TESIS DE MAGISTER

EN INGENIERÍA DEL SOFTWARE

Asistente para la Evaluación de CMMI-SW

AUTOR: ING. MARIO LUIS PERALTA

DIRECTORA

M. ING. PAOLA BRITOS

CO-DIRECTOR

M. ING. EDUARDO DIEZ

BUENOS AIRES, 2004

Dedicatoria

A mi esposa Silvana,
A mis hijos Facundo y Nazarena,
por soportar mis "ausencias" y mis "impaciencias",
por comprenderme y darme su apoyo

A mis queridos padres Iris y Emilio,
por su esfuerzo y su cariño

Resumen

En agosto de 2002, el SEI (*Software Engineering Institute*) liberó el nuevo modelo CMMI (*Capability and Maturity Model Integration*), sucesor del original modelo CMM (*Capability and Maturity Model*).

El nuevo modelo trae asociado un método de evaluación formal llamado SCAMPI (*Standard CMMI Appraisal Method for Process Improvement*), el cual se basa en cuantificar la evidencia encontrada en la organización evaluada y aplicar reglas que permiten inferir los resultados finales de la evaluación.

El sistema desarrollado en este trabajo es un asistente que facilita la evaluación de una organización de acuerdo al modelo CMMI, guiando paso a paso al usuario en la evaluación, y generando automáticamente las valoraciones de acuerdo a las reglas del método SCAMPI.

Abstract

In August of 2002, SEI (*Software Engineering Institute*) released the new CMMI (*Capability and Maturity Model Integration*) model, successor of the original CMM (*Capability and Maturity Model*).

The new model has a method of formal evaluation called SCAMPI (*Standard CMMI Appraisal for Method Process Improvement*), which is based on quantifying the evidence found in the organization under assessment, and applying rules that allow to infer the final evaluation results.

The system developed in this work is an assistant wich facilitates the assessment of an organization according to CMMI model, guiding the user in the evaluation step by step, and automatically generating the valuations according to the rules of SCAMPI.

Tabla de Contenidos

1. INTRODUCCIÓN

1.1 ¿De qué trata esta tesis?

Esta tesis trata sobre el desarrollo de una herramienta de software del tipo "asistente", que facilita la evaluación de una organización de acuerdo al modelo CMMI-SW [CMMI-SW, 2002] y el método SCAMPI [SCAMPI, 2001]. El desarrollo de la misma se basa tecnologías Java y se apoya en la metodología Métrica V3 [Métrica V3, 2000].

El trabajo persigue como objetivo fundamental el de brindar una herramienta que permita acercar el modelo CMMI-SW a las organizaciones de software, y como objetivo secundario el de utilizar la metodología Métrica V3 en un proyecto completo de desarrollo.

1.2 A quiénes está dirigido el texto

El trabajo se encuentra dirigido principalmente a ingenieros de software, profesionales de la industria del software, y cátedras universitarias vinculadas al software.

La documentación contenida en el mismo puede tomarse como referencia para la adopción de prácticas de ingeniería del software o de la metodología Métrica V3, y

como punto de partida para futuras ampliaciones de la herramienta, o eventualmente para el desarrollo de herramientas similares.

1.3 Organización del documento

El material se divide en 13 capítulos que abarcan la totalidad del trabajo de tesis.

- El capítulo **1. Introducción** (el presente) sintetiza los objetivos de la tesis, a quiénes está dirigida, y de qué manera se encuentra organizado el material de la misma.

- El capítulo **2. Introducción al problema** presenta un panorama de la situación actual, mostrando el contexto que da origen al trabajo de tesis.

- El capítulo **3. Estudio de viabilidad del sistema** comprende la documentación resultante del proceso "EVS: Estudio de la Viabilidad del Sistema" de la metodología Métrica V3. La documentación cubre la el análisis de la situación actual, la descripción general del sistema, el estudio de las diferentes alternativas de solución, y la selección de la alternativa más adecuada.

- El capítulo **4. Requerimientos del software y Selección del ciclo de vida** comprende la Especificación de Requerimientos del Software (ERS) y la descripción del modelo de ciclo de vida que se va a aplicar en el proyecto.

- El capítulo **5. Plan general del proyecto** comprende la documentación resultante de las interfaces "Gestión del proyecto", "Gestión de la configuración", "Aseguramiento de la calidad", y "Seguridad", de la metodología Métrica V3. La documentación cubre la estimación del esfuerzo y la planificación del trabajo en todo lo relativo a las interfaces.

- El capítulo **6. Análisis del sistema** comprende la documentación resultante del proceso "ASI: Análisis del Sistema de Información" de la metodología Métrica V3. La documentación cubre el modelado en UML (*Unified Modelling Language*) [Booch & Jacobson & Rumbaugh, 1998] del negocio, los casos de uso, los prototipos de interfaces de usuario, y la estructura y comportamiento del sistema en términos de clases de análisis.

- El capítulo **7. Diseño del sistema** comprende la documentación resultante del proceso "DSI: Diseño del Sistema de Información" de la metodología

Métrica V3. La documentación cubre el modelado en UML del diseño arquitectónico del sistema y los subsistemas que lo conforman, y la estructura y comportamiento del mismo en términos de clases de diseño. Dentro de este capítulo se incluye también el "Plan de pruebas".

- El capítulo **8. Plan de pruebas** comprende la documentación correspondiente a la planificación de las pruebas en sus distintos niveles, y el diseño de todas la pruebas a efectuar en el sistema.

- El capítulo **9. Construcción del sistema** comprende la documentación resultante del proceso "CSI: Construcción del Sistema de Información" de la metodología Métrica V3. La documentación cubre la descripción del entorno de construcción y los resultados de las pruebas unitarias, de integración y del sistema.

- El capítulo **10. Implantación y aceptación del sistema** comprende la documentación resultante del proceso "IAS: Implantación y Aceptación del Sistema" de la metodología Métrica V3. La documentación cubre la incorporación del sistema al entorno productivo y los resultados de las pruebas de implantación y aceptación.

- El capítulo **11. Conclusiones y Futuras líneas de trabajo** contiene las conclusiones obtenidas luego de finalizado el trabajo de tesis, y las futuras líneas de trabajo a seguir por aquellos interesados en el tema.

- El capítulo **12. Bibliografía y Glosario** contiene las referencias bibliográficas utilizadas para el trabajo de tesis, y el glosario con los términos empleados en el mismo.

- El capítulo **13. Anexos** contiene documentación anexa al contenido de la tesis. Entre los anexos figuran la documentación de usuario del sistema, y la documentación relativa a las revisiones efectuadas durante las actividades de aseguramiento de la calidad. Como último anexo, se incorporan las conclusiones finales y las futuras líneas de trabajo.

2. INTRODUCCIÓN AL PROBLEMA

En este capítulo se presentan los conceptos principales del problema a resolver por el presente trabajo de tesis.

El mismo comienza con un breve repaso histórico y una presentación de los conceptos principales. A continuación analiza brevemente la situación actual. Finalmente identifica el problema a resolver y presenta la solución propuesta en el trabajo.

2.1 Historia de CMM

A principios de la década del 80, una firma dedicada al estudio del mercado de Tecnologías de Información publica un reporte [The Standish Group, 2003] sobre el éxito de los proyectos de desarrollo en la industria del software. El reporte, basado en encuestas hechas sobre proyectos de software, informaba los siguientes resultados estadísticos:

- El 30% de los proyectos se cancelaban
- El 54% de los proyectos excedían ampliamente los tiempos y costos estimados
- El 16% de los proyectos finalizaban exitosamente dentro del tiempo, el costo y la funcionalidad prevista

En respuesta a la situación alarmante del momento, el Departamento de Defensa de los EEUU funda el SEI (Instituto de Ingeniería del Software, en inglés *Software Engineering Institute*) [SEI, 2003] en la universidad Carnegie Mellon, con el propósito de estudiar el problema y encontrar alguna solución.

En 1991, el SEI publica el modelo CMM (Modelo de Madurez de Capacidad, en inglés *Capability Maturity Model*) [CMM, 1991]. El modelo está orientado a la mejora de los procesos relacionados con el desarrollo de software, para lo cual contempla las consideradas mejores prácticas de ingeniería de software y de gestión.

A partir de ese momento, el Departamento de Defensa exige que sus proveedores estén acreditados en CMM, lo que impulsa a que el modelo tenga una amplia aceptación y se convierta en un estándar de facto dentro de la industria del software.

El modelo CMM original define cinco niveles de madurez dentro de los cuales se puede encontrar una organización [CMM, 1991]:

- *Nivel 1 – Inicial*: el proceso de software es impredecible, sin control y reactivo. El éxito de los proyectos depende del talento de los individuos.
- *Nivel 2 – Repetible*: existen procesos básicos de gestión los proyectos (costo, calendario, funcionalidad). Los procesos existentes hacen que se puedan repetir éxitos en proyectos de similares características.
- *Nivel 3 – Definido*: existe un proceso de software documentado y estandarizado dentro de la organización. Todos los proyectos utilizan una versión a medida del proceso.
- *Nivel 4 – Gestionado*: la organización recolecta métricas del proceso software y de los productos desarrollados. Tanto el proceso como los productos se entienden y controlan cuantitativamente.
- *Nivel 5 – Optimizado*: existe una mejora continua del proceso software, basada en la realimentación cuantitativa del proceso y en la puesta en práctica de ideas y tecnologías innovadoras.

De acuerdo al modelo, el rendimiento en general de una organización mejora notablemente a medida que la misma incrementa su nivel de madurez. La figura 2.1 muestra de manera conceptual las mejoras en el rendimiento para cada nivel contemplando los factores Tiempo y Costo [Paulk *et al*, 1993]. Conclusiones similares se pueden extraer para otros factores como la Funcionalidad y la Calidad.

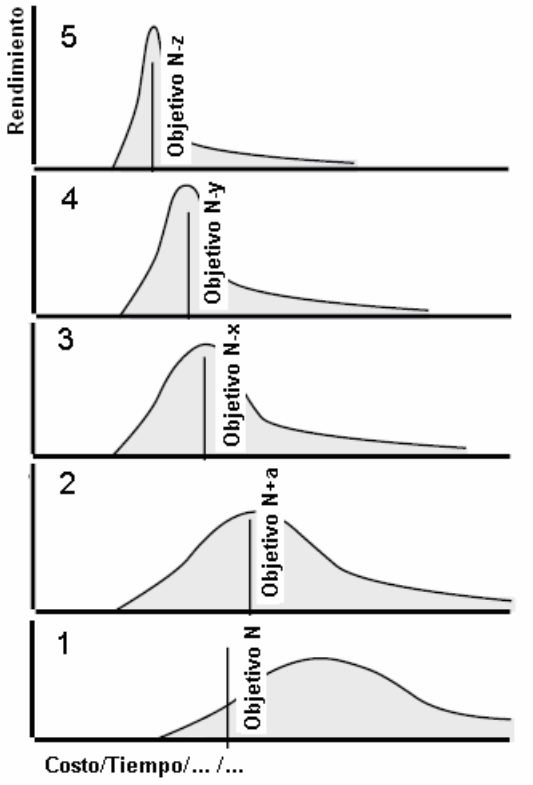

Figura 2.1. Variación del rendimiento con respecto a los objetivos fijados para los factores costo y tiempo, de acuerdo al nivel de CMM de la organización [Paulk *et al*, 1993].

En la misma se puede observar lo siguiente:

- En las organizaciones que se encuentran en el nivel 1, los objetivos generalmente son ampliamente excedidos por la realidad.
- En las organizaciones que se encuentran en el nivel 2, se establecen objetivos más acordes a la realidad.
- En las organizaciones que se encuentran los niveles 3, 4, y 5, existe una menor dispersión de la realidad con respecto a los objetivos, y la performance mejora con cada nivel.

Algunas de las organizaciones que adoptaron el modelo fueron [SEIR, 2003]: Accenture, AT&T, Boeing, Ericcson, Fuji Xerox, Hewlett Packard, Hyundai, IBM, Motorola, Nasa, NCR, NEC, PriceWaterhouseCoopers, Samsung, Siemens, United Airlines, entre otras.

Luego del éxito alcanzado por CMM, el SEI desarrolló modelos similares para otras disciplinas, entre las cuales figuraban la ingeniería de sistemas (SE-CMM, *Systems Engineering Capability Maturity Model*), la adquisición de software (SA-CMM, *Software Acquisition Capability Maturity Model*), las personas (P-CMM, *People*

Capability Maturity Model), y el desarrollo integrado de productos (IPD-CMM, *Integrated Product Development Capability Maturity Model*) [CMMS, 2003].

A mediados de la década del 90, el SEI decide unificar los modelos de ingenieria de software (SW-CMM, también conocido como CMM), de ingeniería de sistemas (SE-CMM) y de desarrollo integrado de productos (IPD-CMM), embarcándose en un esfuerzo que culmina en el año 2002 dando origen a una nueva generación llamada CMMI (Integración del Modelo de Madurez de Capacidad, en inglés *Capability Maturity Model Integration*) [CMMI, 2002].

El nuevo modelo CMMI brinda un marco con una estructura común para todas las disciplinas (ingeniería de software, ingeniería de sistemas, desarrollo integrado de productos, adquisición de productos) y agrega una nueva forma de representación además de la conocida representación por niveles. La nueva forma de representación se llama *Continua* y está orientada a medir la mejora en los procesos de manera individual en vez de hacerlo de manera conjunta como la representación por niveles [Chrissis & Konrad & Shrum, 2003].

Dentro de esta nueva generación de modelos, el sucesor directo del CMM original es el denominado CMMI-SW [CMMI-SW, 2002]. Este modelo presenta una mayor cobertura con respecto a las áreas de proceso, y agrega el concepto de representación continua.

En paralelo con el desarrollo de CMMI, el SEI elaboró un método para la evaluación formal del modelo denominado SCAMPI (Método de Evaluación Estándar de CMMI para la Mejora de Procesos, en inglés *Standard CMMI Appraisal Method for Process Improvement*) [SCAMPI, 2001].

El método SCAMPI consta de tres fases, en cada una de las cuales se llevan a cabo un conjunto de procesos. La tabla 2.1 resume las fases y procesos de SCAMPI.

Los resultados de una evaluación se obtienen mediante la aplicación de un conjunto de reglas de negocio aplicadas a cada componente del modelo (*prácticas*, *objetivos*, *áreas de proceso* y *niveles de madurez*). Estas reglas hacen que sea necesario utilizar herramientas, ya que el método de evaluación deja de ser una simple encuesta para convertirse en una evaluación detallada y casi matemática.

La situación actual con respecto a los modelos es la siguiente:

- El SEI ha iniciado la discontinuación gradual del modelo CMM original (más conocido como SW-CMM) en diciembre del 2003, para finalizarla en el 2005. Ante esta situación, sugiere a todas las organizaciones acreditadas migrar a CMMI-SW [SW-CMM Migration, 2001].
- Las grandes organizaciones acreditadas en SW-CMM planean migrar a CMMI-SW [CMMI Adoption, 2003].

- Muchas organizaciones pequeñas planean acreditarse en CMMI-SW, con el fin de poder acceder al mercado de las exportaciones.
- La preparación previa a la acreditación CMMI-SW es larga y costosa. Las organizaciones utilizan el concepto de "Evaluación interna" como paso preparatorio [Motorola, 2003].
- Una "Evaluación interna" es algo difícil de llevar a cabo para las organizaciones recién iniciadas en el tema, y no existe un soporte adecuado de herramientas que le faciliten el camino [Motorola, 2003].

Fase	***Proceso***	***Propósito***
1. Planificación y preparación para la evaluación	1.1 Analizar requerimientos	Entender las necesidades de negocio de la organización. Nivelar los objetivos del negocio con los objetivos de la evaluación.
	1.2 Desarrollar plan de evaluación	Documentar requerimientos, acuerdos, estimaciones, riesgos, personalizaciones del método y consideraciones prácticas. Consensuar el plan de evaluación con la organización.
	1.3 Seleccionar y preparar equipo	Asegurar que un equipo calificado esté a cargo de la ejecución de la evaluación.
	1.4 Obtener y analizar evidencia objetiva inicial	Obtener información que facilite la preparación de la evaluación. Identificar potenciales fortalezas y debilidades. Obtener un entendimiento preliminar de las operaciones y procesos de la organización.
	1.5 Preparar la recolección de evidencia objetiva	Planificar y documentar las estrategias para la recolección de datos, incluyendo fuentes de datos, herramientas y tecnologías a utilizar.
2. Conducción de la evaluación	2.1 Examinar la evidencia objetiva	Recolectar información sobre las prácticas implementadas en la organización, siguiendo el plan de recolección definido.
	2.2 Verificar y validar la evidencia objetiva	Verificar la implementación de las prácticas en la organización. Cada práctica implementada se compara con la definición del modelo CMMI, y el equipo le asigna una valoración.
	2.3 Documentar la evidencia objetiva	Crear registros que documenten la implementación de las prácticas, contemplando también las fortalezas y debilidades encontradas.
	2.4 Generar los resultados de la evaluación	Calificar la satisfacción de los objetivos de acuerdo a las valoraciones asignadas a las prácticas. Calificar la satisfacción de las áreas de proceso de acuerdo a la satisfacción de los objetivos. Calificar los niveles de capacidad o madurez de acuerdo a la satisfacción de las áreas de proceso.
3. Reporte de los resultados	3.1 Entregar los resultados de la evaluación	Entregar los resultados obtenidos a la organización, de manera que puedan ser utilizados para tomar acciones futuras.
	3.2 Empaquetar y archivar los activos de la evaluación	Preservar los datos y registros importantes resultantes de la evaluación, almacenándolos de manera apropiada.

Tabla 2.1. Fases y procesos del método SCAMPI.

En síntesis, el nuevo modelo trae aparejado un problema no trivial para las organizaciones, en lo referente a los costos y tiempos necesarios para la preparación previa a su adopción o a una acreditación. El problema se ve más acentuado en las organizaciones pequeñas, donde los recursos económicos, humanos y temporales suelen ser menores que en las grandes organizaciones.

En este sentido, sería deseable contar con una herramienta que asista a las organizaciones en la conducción de una evaluación interna, indicando paso a paso los aspectos del modelo CMMI y los criterios de evaluación del método SCAMPI, de manera de disminuir los costos y tiempos necesarios para la preparación previa a una acreditación.

2.2 Estado de la tecnología

Existen actualmente tres herramientas de evaluación para CMMI:

- *CMM-Quest*: permite efectuar evaluaciones de acuerdo al modelo CMMI-SE/SW en su representación continua. La evaluación se limita a asignar valores a los objetivos, no permite evaluaciones a nivel de prácticas (por debajo del nivel de los objetivos). No brinda soporte para el método SCAMPI [CMM-Quest, 2001].
- *IME Toolkit*: permite efectuar evaluaciones de acuerdo al modelo CMMI-SE/SW. Las evaluaciones consisten en asignar valores numéricos a las prácticas, en base a los cuales la herramienta genera puntajes para las áreas de proceso. No brinda soporte para el método SCAMPI. No posee guías de asistencia para la evaluación [IME Toolkit, 2003].
- *Appraisal Wizard*: soporta evaluaciones para gran parte de los modelos CMM y métodos de evaluación propuestos por el SEI a lo largo de la historia (entre ellos, todos los CMMI y SCAMPI). Está pensada para cubrir todas las necesidades del método SCAMPI, requiriendo amplios conocimientos del mismo por parte del usuario. Requiere que el usuario ingrese todos los valores que se asignan en las distintas instancias de evaluación (prácticas, objetivos, áreas de proceso) y no cuenta con la capacidad de sugerir valores facilitando las tareas de ingreso de datos. Al brindar un soporte tan amplio y detallado, la herramienta no es para nada sencilla de utilizar [Appraisal Wizard, 2003].

La tabla 2.2 muestra en forma comparativa las características de estas herramientas.

Analizando la tabla se desprende que existe un área no soportada por las herramientas existentes. Esta área está conformada principalmente por características orientadas a los usuarios novatos, como la navegación de la estructura del modelo, las guías paso a paso, la generación automática de valoraciones, y la selección del nivel de granularidad para la evaluación.

	CMM-Quest	*IME Toolkit*	*Appraisal Wizard*
Interfaz de usuario	Fácil, muy amigable	Medianamente amigable	Difícil, poco amigable
Tipo de usuario	Novato	Experto	Experto
Modelos soportados	CMMI-SE/SW (representación Continua)	Está basada en CMMI-SE/SW, no lo soporta formalmente	Gran parte de los CMM y todos los CMMI (ambas representaciones)
Método SCAMPI	No	No	Sí
Nivel de granularidad	Grueso (sólo objetivos)	Fino (hasta prácticas específicas)	Fino (hasta prácticas específicas)
Ayudas online	Sí	No	Sí
Navegación de la estructura del modelo	No	No	No
Generación de valores sugeridos	No	Si	No
Selección del nivel de granularidad para la evaluación	No	No	No

Tabla 2.2. Comparación de características de herramientas

2.3 Identificación del problema

La preparación que debe llevar a cabo una organización como paso previo a la adopción del modelo CMMI-SW o a su acreditación, es un proceso largo, costoso, y requiere de recursos humanos altamente capacitados. Algunas organizaciones llevan a cabo una evaluación interna como parte del proceso de preparación.

En la actualidad no existen herramientas que brinden un soporte adecuado para la conducción de evaluaciones internas basadas en el método de evaluación SCAMPI. Las herramientas existentes o bien son demasiado informales y no soportan el método, o bien están orientadas a usuarios profesionales y no resultan amigables a los usuarios novatos.

El objetivo principal del trabajo de tesis es desarrollar una herramienta que brinde la posibilidad de efectuar evaluaciones internas a las organizaciones interesadas en el modelo CMMI. La herramienta contará con características orientadas a usuarios no expertos de manera de facilitarles el entendimiento del modelo y su método de evaluación. Entre las características previstas para la herramienta, figuran las siguientes:

- Soporte del modelo CMMI-SW en su representación por niveles.
- Soporte del método de evaluación SCAMPI.
- Selección del nivel de granularidad para la evaluación.
- Navegación sobre la estructura del modelo, con guías que expliquen al usuario los criterios que deben seguirse para la evaluación en cada parte de la estructura.
- Generación automática de valoraciones de acuerdo a las reglas del método SCAMPI.
- Generación de reportes con las valoraciones y las conclusiones finales de la evaluación.

El alcance de la herramienta se limita al modelo CMMI-SW en su representación por niveles. Se ha seleccionado esta alternativa debido a que este modelo es el sucesor directo del SW-CMM original, y la representación por niveles posibilita la comparación directa entre evaluaciones efectuadas en distintas organizaciones.

Como fuente de información para el trabajo de tesis de tomarán las especificaciones del modelo CMMI-SW [CMMI-SW, 2002] y del método SCAMPI [SCAMPI, 2001] provistas por el SEI.

2.4 Solución propuesta

La herramienta a desarrollar como objetivo del trabajo de tesis será una aplicación del tipo *standalone*, ejecutable en cualquier sistema operativo con soporte para la plataforma Java (esto incluye todas las variantes de Windows, gran parte de los sistemas Unix, Linux, FreeBSD y otros OpenSource).

La misma contará con una interfaz gráfica de navegación donde se muestre un panorama general del modelo CMMI-SW. En esa interfaz se presentarán los componentes del modelo en el mismo orden en que se encuentran en la especificación CMMI-SW, es decir, *niveles de madurez*, *áreas de proceso*, *objetivos*, y *prácticas*. En cada componente, la herramienta presentará una guía para asistir al usuario en la evaluación del mismo.

Durante la evaluación, la herramienta permitirá que el usuario asigne valores a las distintas *prácticas*. Sobre los valores asignados extraerá conclusiones de acuerdo a las reglas definidas en el método SCAMPI, las cuales serán presentadas como opciones para la valoración de los *objetivos* y las *áreas de proceso*. El usuario podrá seguir las conclusiones de la herramienta como guía para la asignación de valoraciones.

Luego de finalizadas las distintas instancias de evaluación, permitirá obtener un reporte general del nivel CMMI de la organización o proyecto, incluyendo las valoraciones efectuadas por el usuario.

Con respecto a las fases y procesos del método SCAMPI, la herramienta brindará soporte para la ejecución de la fase 2 (Conducción de la evaluación) y parte de la fase 3 (Reporte de los resultados).

Como soporte de la fase 2, brindará guías para verificar la implementación de las prácticas, permitirá asignar valoraciones y documentar las fortalezas y debilidades, y generará resultados sugeridos para la evaluación aplicando las reglas del método.

Como soporte de la fase 3, permitirá generar reportes para entregar los resultados.

3. ESTUDIO DE VIABILIDAD DEL SISTEMA

En este capítulo se documentan los productos de salida del proceso EVS (Estudio de Viabilidad del Sistema) de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

3.1 Descripción general del sistema

3.1.1 Necesidades del usuario

La necesidad principal planteada por el usuario se traduce en una herramienta de software que permita evaluar a una organización o proyecto de acuerdo a las distintas instancias del modelo CMMI-SW [CMMI-SW, 2002] en su representación por niveles, guiando al usuario en todo momento para facilitar la evaluación. La herramienta a construir deberá estar basada en el método SCAMPI [SCAMPI, 2001] para la evaluación de CMMI [CMMI, 2002].

Para cubrir esta necesidad, la herramienta debe contar con una interfaz gráfica donde se muestre un panorama general del modelo. A partir de esa interfaz, el usuario podrá navegar sobre las distintas instancias de evaluación del modelo (áreas de proceso, objetivos, prácticas) asignando valores a las mismas de acuerdo a *guías* y *sugerencias* provistas por la herramienta.

Las *guías* deben contener la documentación del modelo CMMI-SW completa, contemplando las áreas de proceso, sus objetivos específicos y genéricos, y las prácticas que permiten evaluar esos objetivos.

Las *sugerencias* deben presentarse de acuerdo a las reglas definidas en el método SCAMPI, esto es, de acuerdo a los valores asignados por el usuario a las diferentes prácticas, la herramienta utilizará las reglas de SCAMPI para sugerir valores a asignar en los objetivos; de la misma manera, de acuerdo a los valores de los objetivos puede sugerir valores para las áreas de proceso; finalmente, de acuerdo a los valores de las áreas de proceso, sugerirá valores para el nivel de madurez.

Luego de finalizadas las distintas instancias de evaluación, la herramienta permitirá obtener un reporte general del nivel CMMI de la organización o proyecto, en el cual se incluyan las valoraciones efectuadas por el usuario.

3.1.2 Restricciones

En el orden económico, no existen restricciones que afecten al desarrollo de la herramienta. La misma puede plantearse sobre plataformas y entornos de desarrollo OpenSource, concebida como una herramienta de escritorio y sin necesidad de adquirir software comercial de terceras partes. Por otra parte, no es necesario prever recursos de hardware o humanos, ya que se trata de un trabajo de tesis elaborado por completo en la computadora personal del tesista.

Tampoco existen restricciones de carácter técnico que afecten al desarrollo de la herramienta. Prueba de ello es la amplia variedad de herramientas de escritorio que existen en el mercado, OpenSource y comerciales, que resuelven problemas parecidos en otras ramas (entornos de desarrollo, procesadores de texto, planillas de cálculo, navegadores web, navegadores del sistema de archivos, etc).

Finalmente, tampoco existen restricciones en el orden legal. El SEI (*Software Engineering Institute*) [SEI, 2003] publica los modelos CMMI y el método SCAMPI de manera gratuita, y no presenta requisitos legales para su utilización como referencia.

Ante este panorama, las alternativas de solución son básicamente las siguientes:

- Encontrar una herramienta que cubra las necesidades del usuario
- Desarrollar una herramienta basándose en dichas necesidades

Como parte del estudio de viabilidad, se hará un análisis comparativo de las herramientas existentes, para finalmente contrastarlo con la segunda opción. En función de los resultados obtenidos, se decidirá si conviene desarrollar una herramienta nueva o no.

Cabe aclarar que en el caso de optar por la segunda opción (desarrollo de la herramienta), las diferentes alternativas para el desarrollo pasan fundamentalmente por los lenguajes de programación y las herramientas a utilizar, llevando todas ellas a la misma solución que es la herramienta de escritorio desarrollada como un sistema informático convencional.

Por este motivo, no se evaluará cuál es el lenguaje o el conjunto de herramientas más adecuado para el desarrollo, sino que simplemente se seleccionará alguna opción dentro de las alternativas OpenSource.

3.2 Catálogo de objetivos del Estudio de Viabilidad

Teniendo en cuenta el análisis de las necesidades del usuario y las restricciones existentes efectuado en el apartado anterior, los objetivos principales del estudio de viabilidad son los siguientes:

- Analizar la situación actual
- Describir las posibles mejoras
- Analizar las alternativas de solución
- Establecer los requerimientos del nuevo sistema
- Planificar el desarrollo del nuevo sistema

3.3 Actividades a realizar en el Estudio de Viabilidad

Las actividades que se llevarán a cabo para el Estudio de la Viabilidad del Sistema son las siguientes:

- EVS1 Establecimiento del alcance del sistema
- EVS2 Estudio de la situación actual
- EVS3 Definición de requisitos del sistema
- EVS4 Estudio de las alternativas de solución
- EVS5 Valoración de las alternativas
- EVS6 Selección de la solución

En lo que respecta a las interfaces, se llevarán a cabo las siguientes actividades en paralelo con el estudio de viabilidad:

- *Gestión de proyectos*: a realizar una vez finalizado el estudio de viabilidad
    - GPI1 Estimación del esfuerzo
    - GPI2 Planificación

- *Gestión de la configuración*: a realizar durante el estudio de la viabilidad y una vez finalizado el mismo
    - EVS-GC1 Definición de los requisitos de gestión de configuración
    - EVS-GC2 Establecimiento del plan de gestión de la configuración
- *Aseguramiento de la calidad*: a realizar durante el estudio de la viabilidad y una vez finalizado el mismo
    - EVS-CAL1 Identificación de las propiedades de calidad para el sistema
    - EVS-CAL2 Establecimiento del plan de aseguramiento de la calidad
- Seguridad: debido a las características del sistema (standalone y monousuario), y a las características del proyecto (tesis llevada a cabo por una única persona), las actividades de seguridad se limitan a lo mínimo indispensable.
    - EVS-SEG5 Evaluación detallada de la seguridad de la solución propuesta

El plan de trabajo para la ejecución de las actividades mencionadas, y los productos resultantes de las mismas, se encuentran documentados en el Capítulo 5 (Plan general del proyecto).

El esquema general de las actividades se puede ver en la figura 3.1.

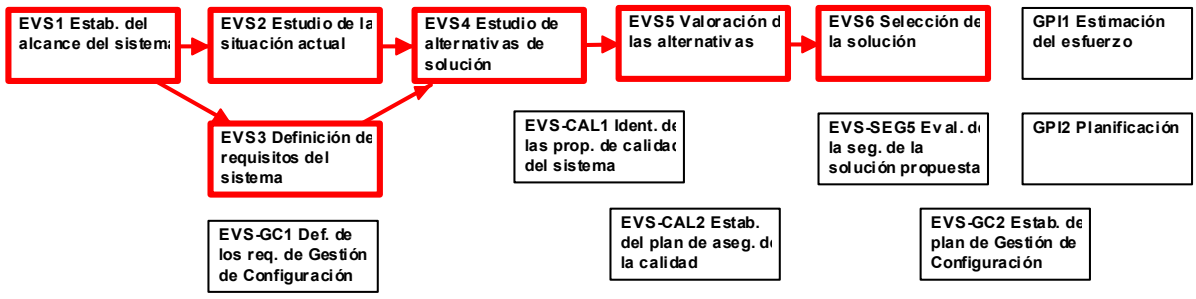

Figura 3.1: esquema general de actividades para el estudio de viabilidad del sistema, incluyendo interfaces.

3.4 Descripción de la situación actual

Como resultado de algunas búsquedas efectuadas en Internet, se encontraron dos herramientas de características similares a las planteadas por el usuario. Debido a esto, se realiza un estudio de las mismas con el fin de establecer el alcance y los requerimientos de la herramienta planteada como objetivo de la tesis. El estudio de las herramientas existentes se limita a analizar las características principales de cada una de ellas y su correspondencia con el modelo CMMI-SW y el método SCAMPI.

Las herramientas a analizar son las siguientes:

- *CMM-Quest*: desarrollada por HM&S IT Consulting [CMM-Quest, 2001].
- *Appraisal Wizard*: desarrollada por Integrated System Diagnostics Incorporated [Appraisal Wizard, 2003].

Cabe destacar que se encontró una tercer herramienta llamada *IME Toolkit* [IME Toolkit, 2003], ya mencionada en el capítulo 2. Sin embargo, no se efectúa una evaluación detallada de la misma debido a que no es una herramienta propiamente dicha, sino un conjunto de planillas Excel que no posee ninguna de las características deseadas.

3.4.1 CMM-Quest

Es una herramienta pensada para asistir al usuario en la autoevaluación de una organización de software. El objetivo principal de la misma es el de dar soporte a las evaluaciones basadas en métodos informales (clase B y C[1]). No soporta el método SCAMPI (clase A).

Para las evaluaciones, soporta el modelo CMMI-SE/SW/IPPD/SS [CMMIS, 2002] en su representación continua. Sin embargo, se limita a analizar las áreas de proceso con sus objetivos, sin llegar al análisis de las prácticas que permiten evaluar estos últimos.

Las características principales de la herramienta se pueden dividir en los siguientes grupos:

- *Configuración*: permite configurar el alcance de la evaluación, indicando los objetivos de las áreas de proceso que se van a incluir.
- *Evaluación*: permite evaluar los objetivos de manera visual, con barras que indican el porcentaje de satisfacción de los mismos.
- *Reportes gráficos*: posee una amplia variedad de gráficos para presentar los resultados de la evaluación.

[1] El SEI presenta tres clases de métodos de evaluación:

- Clase A: es el más completo, con resultados más exactos, brindando un mayor entendimiento de las fortalezas y debilidades de la organización. El único exponente de esta clase es el método SCAMPI.
- Clase B: es un método de menor escala, también llamado mini-appraisal. Se entra en menos detalles que en el clase A, y requiere un menor esfuerzo.
- Clase C: es el menos intensivo de los tres, también llamado micro-appraisal. Sirve para tener una idea rudimentaria de las prácticas efectuadas en una organización.

El único método avalado formalmente por el SEI es SCAMPI (clase A). Los métodos clase B y C no tienen una especificación formal por parte del SEI, quedando su implementación a cargo de los interesados.

- *Almacenamiento*: permite almacenar las evaluaciones en archivos y recuperarlas posteriormente para su edición.
- *Asistencia al usuario*: posee ayudas online y guías para la evaluación.

Las figuras 3.2 a 3.5 muestran algunos ejemplos de las características mencionadas.

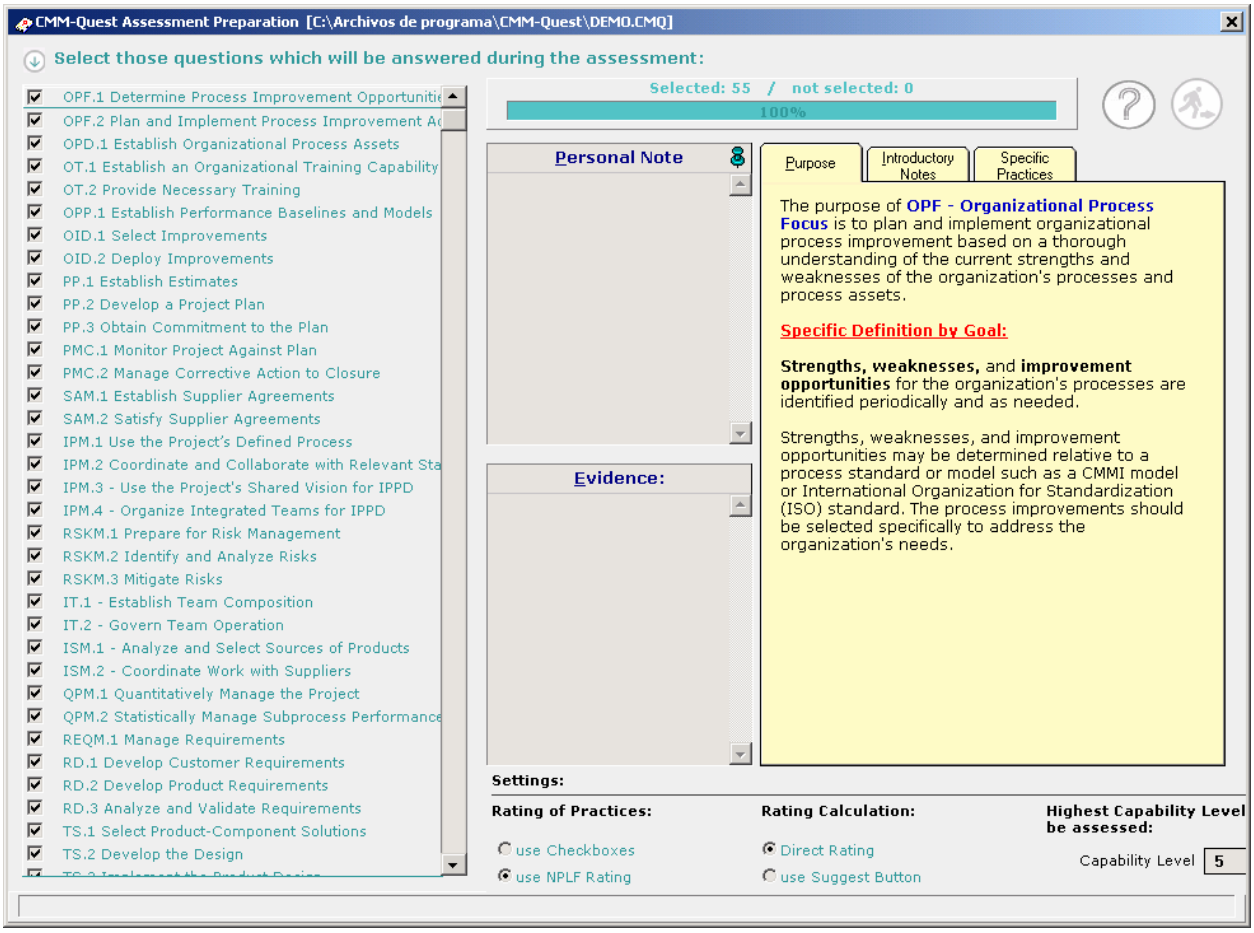

Figura 3.2. Configuración de los objetivos a evaluar.

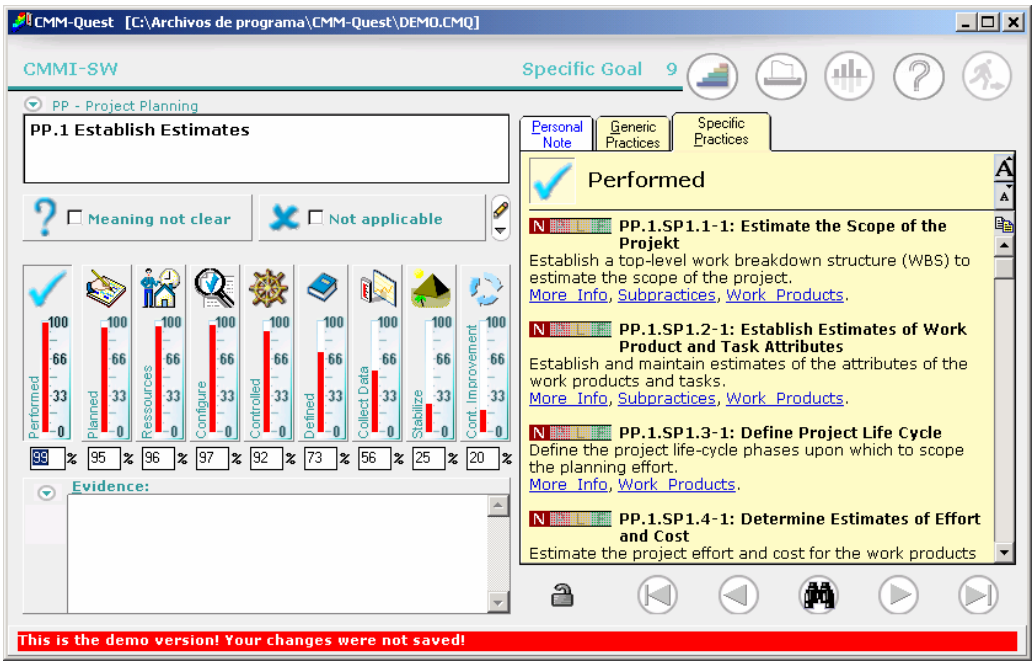

Figura 3.3. Evaluación de uno de los objetivos (PP.1 Establish Estimates) del área de proceso PP – Project Planning. En la parte derecha se pueden observar las guías y ayudas para asistir al usuario en la evaluación. En la parte izquierda se puede ver el área para asignación de valores de manera visual.

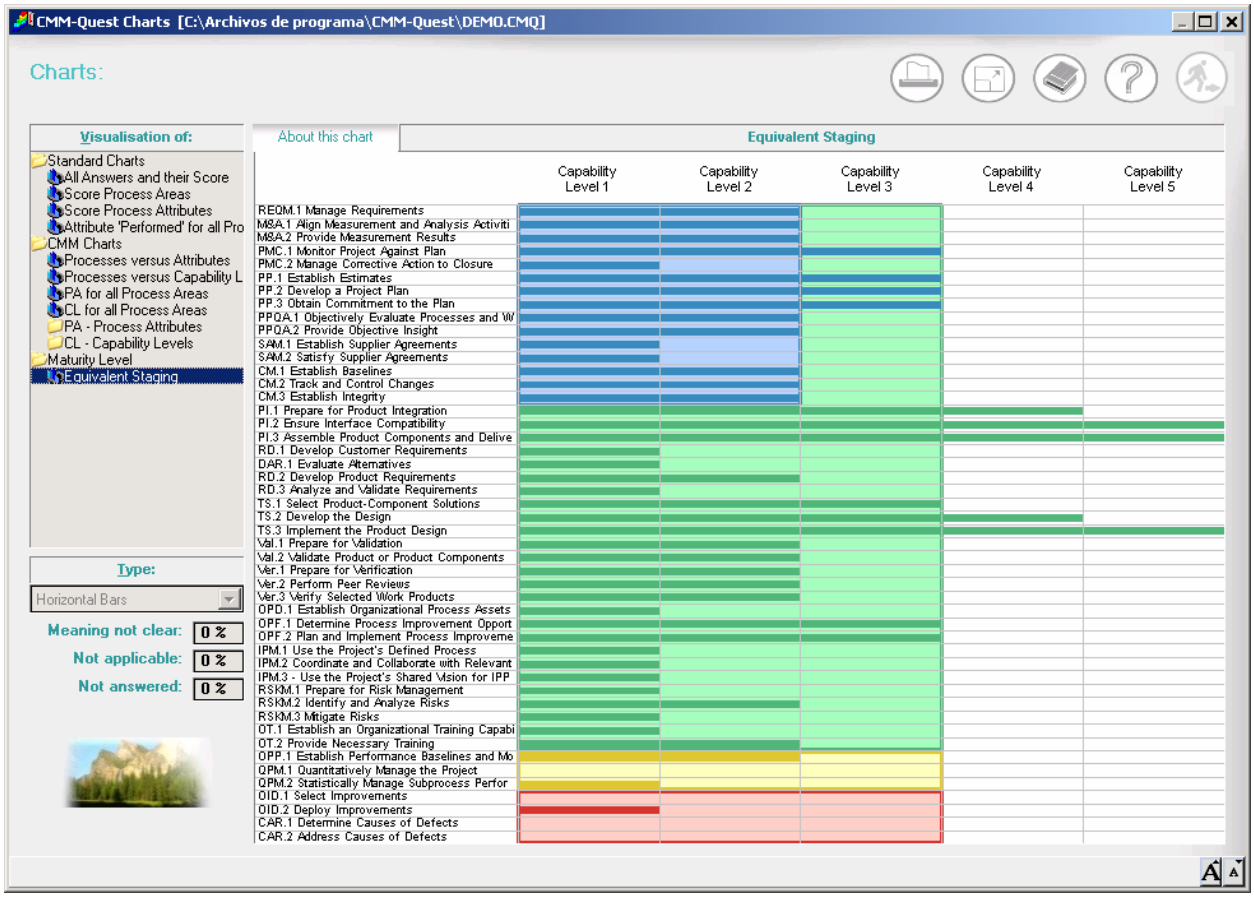

Figura 3.4: gráfico que muestra las equivalencias entre los objetivos y los niveles de capacidad.

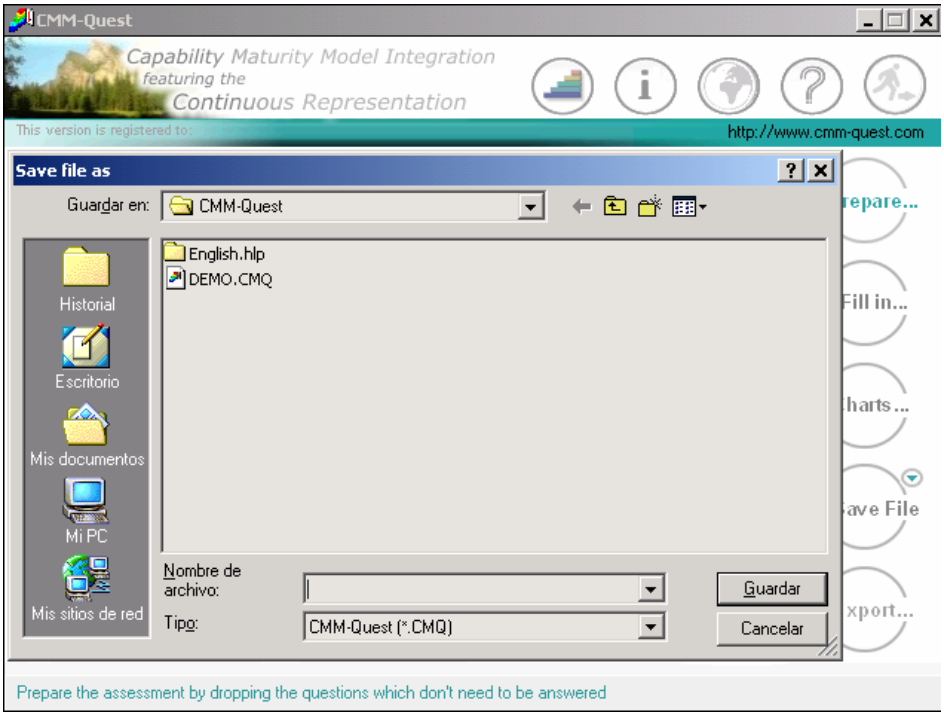

Figura 3.5. Almacenamiento de una evaluación en archivo.

Por último, cabe mencionar los requerimientos de software y hardware de la herramienta:

- Sistema operativo: Windows 95/98/ME or NT 4.0/2000/XP
- Espacio en disco: 20 MB
- No tiene requerimientos especiales de memoria o video
- No requiere software de terceras partes

3.4.2 Appraisal Wizard

Es una herramienta pensada para asistir a un equipo de personas en las tareas de evaluación de una organización. Tiene en cuenta la planificación del trabajo, la recopilación de datos, las valoraciones efectuadas por el equipo, y la generación de los resultados que deben entregarse a la organización como conclusiones de la evaluación.

Soporta prácticamente todos los modelos publicados por el SEI, incluyendo el modelo CMMI-SE/SW/IPPD/SS en ambas representaciones (continua y por niveles). Además, da soporte a varios métodos de evaluación, entre ellos SCAMPI.

Las características principales de la herramienta se pueden dividir en los siguientes grupos:

- *Configuración*: permite definir usuarios, valores por defecto en tablas, y formato de los reportes por defecto.
- *Planificación*: permite registrar información de la organización bajo análisis y de los proyectos a analizar como parte de la evaluación. Asimismo, permite definir los miembros del equipo de evaluación y el calendario de actividades.
- *Recopilación de datos*: permite ingresar observaciones sobre las fortalezas y debilidades de la organización encontradas durante la evaluación. Las observaciones pueden ser vinculadas a las prácticas del modelo. Además de esto, permite evaluar cada una de las prácticas y objetivos del modelo.
- *Análisis de cobertura y valoraciones*: brinda un conjunto de interfaces que consolidan los datos ingresados, de manera que el equipo de evaluación pueda asignar valores a las diferentes instancias de evaluación (prácticas, objetivos, áreas de proceso).
- *Reportes y gráficos*: permite generar reportes y gráficos con los valores asignados por el equipo de evaluación. Incluye reportes y gráficos sobre las prácticas, los objetivos, las áreas de proceso y los niveles de madurez.
- *Almacenamiento*: la herramienta almacena toda la información en una base de datos.
- *Asistencia al usuario*: posee ayudas online y guías para las diferentes etapas de la evaluación.

Además de estas características, incluye facilidades de importación y exportación de datos, un corrector ortográfico para las observaciones, y un navegador de objetivos y prácticas por áreas de proceso.

Las figuras 3.6 a 3.13 muestran algunos ejemplos de las características mencionadas.

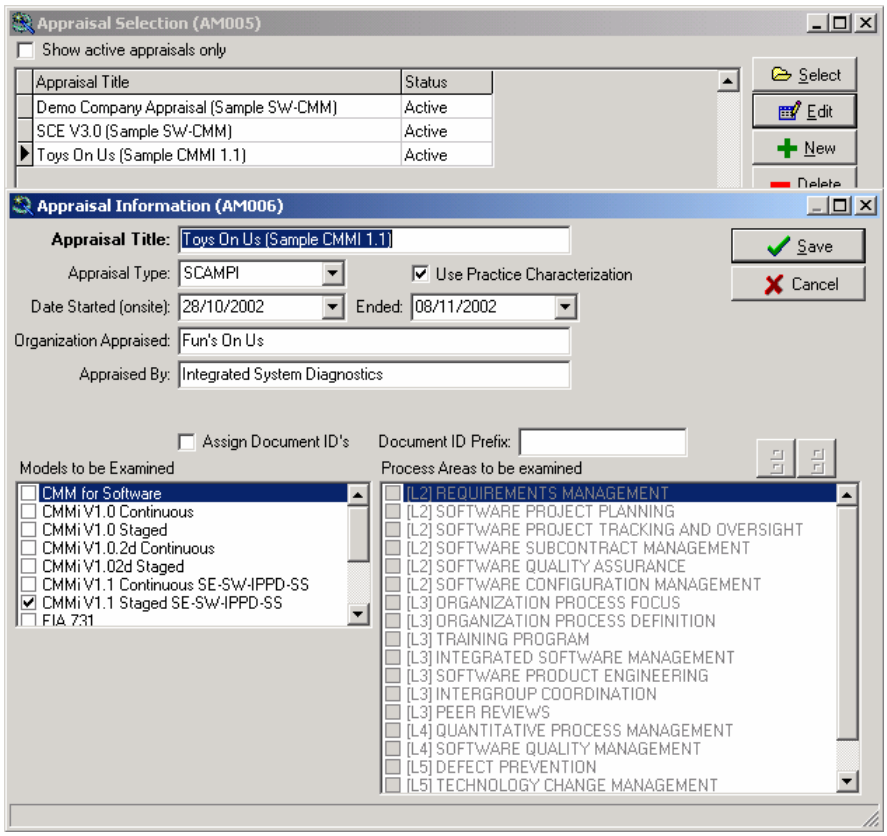

Figura 3.6. Selección de evaluaciones y edición de la información asociada a una evaluación.

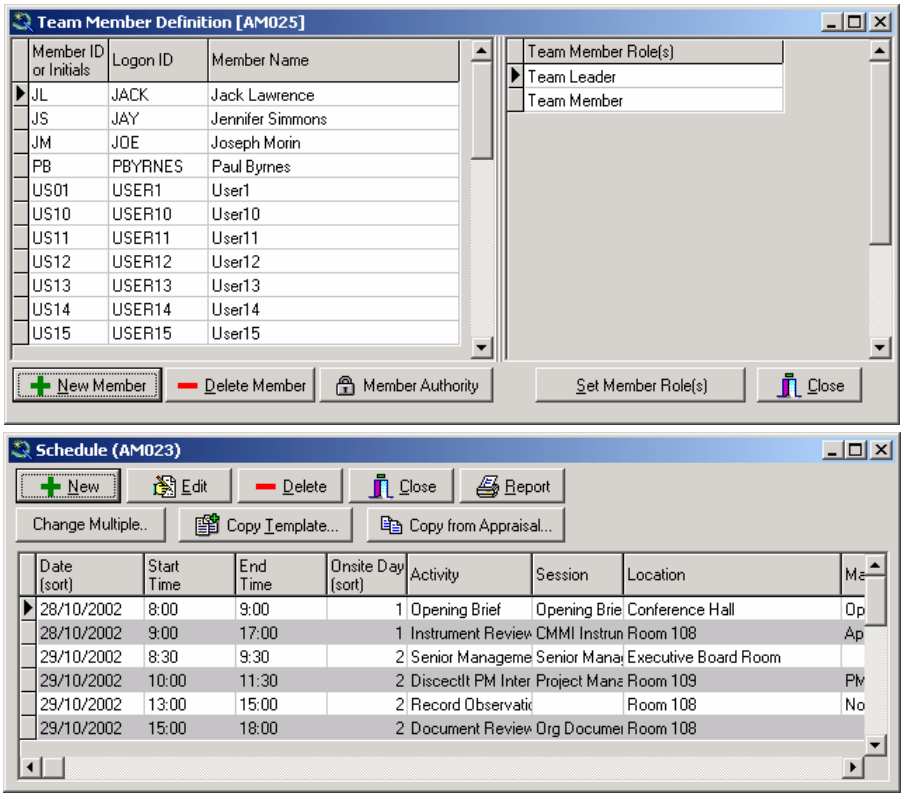

Figura 3.7. Planificación de integrantes y Calendario de evaluación.

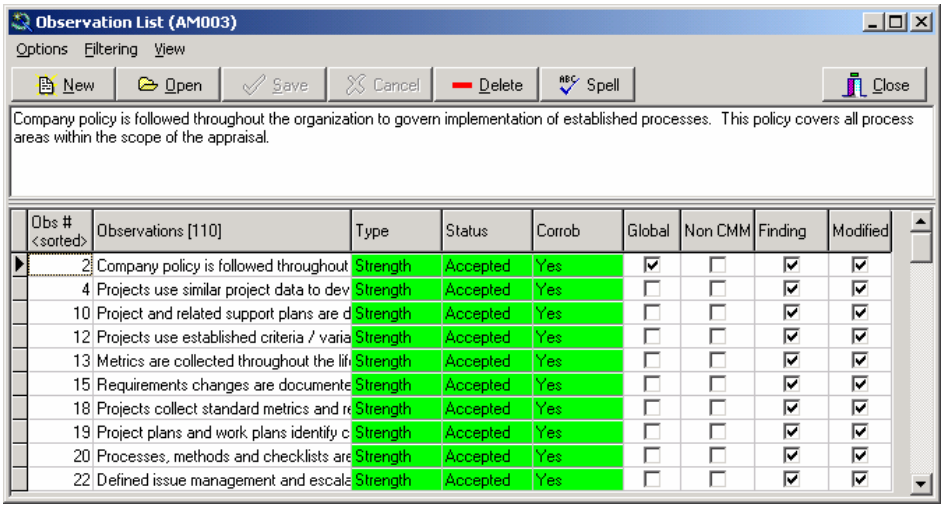

Figura 3.8. Administración de observaciones sobre fortalezas y debilidades.

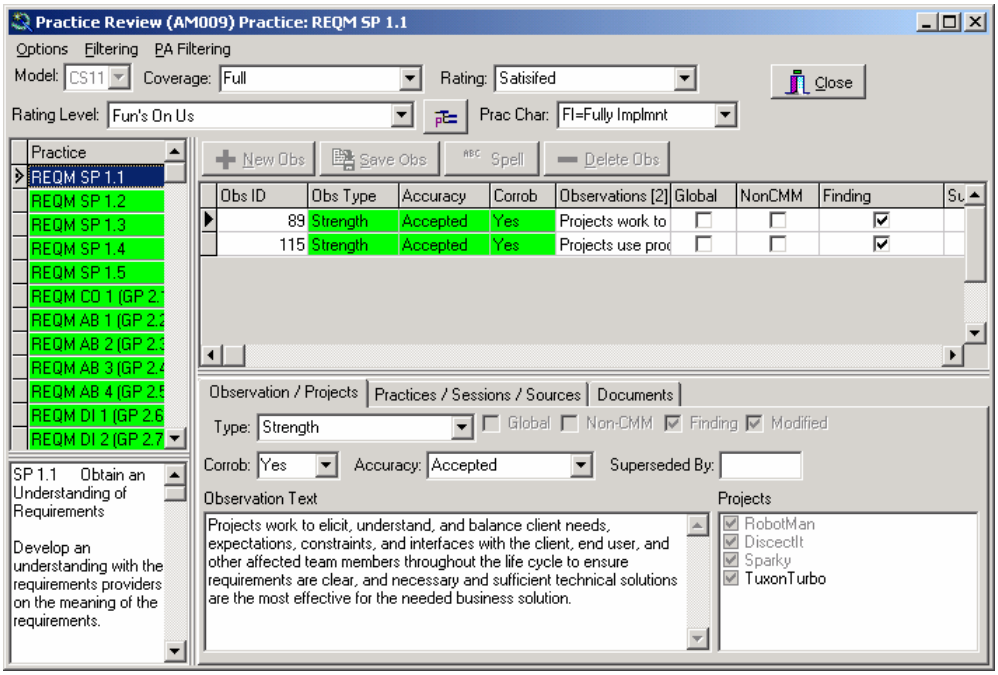

Figura 3.9. Revisión de prácticas. En el ejemplo pueden observarse las opciones de valorización, las guías, y el vínculo entre las observaciones y las prácticas.

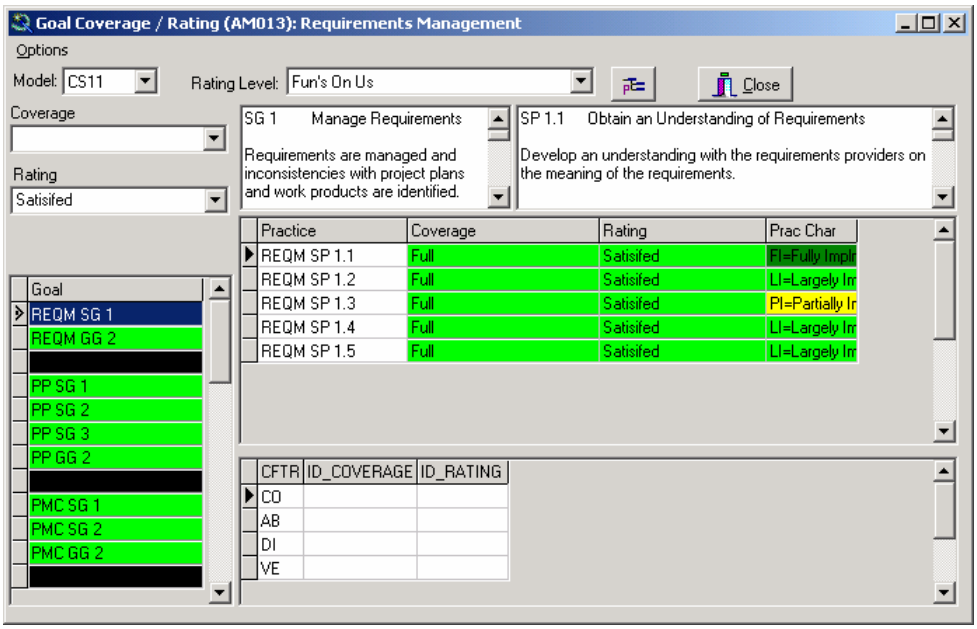

Figura 3.10. Revisión de objetivos. En el ejemplo pueden observarse las opciones de valorización, las guías de asistencia al usuario.

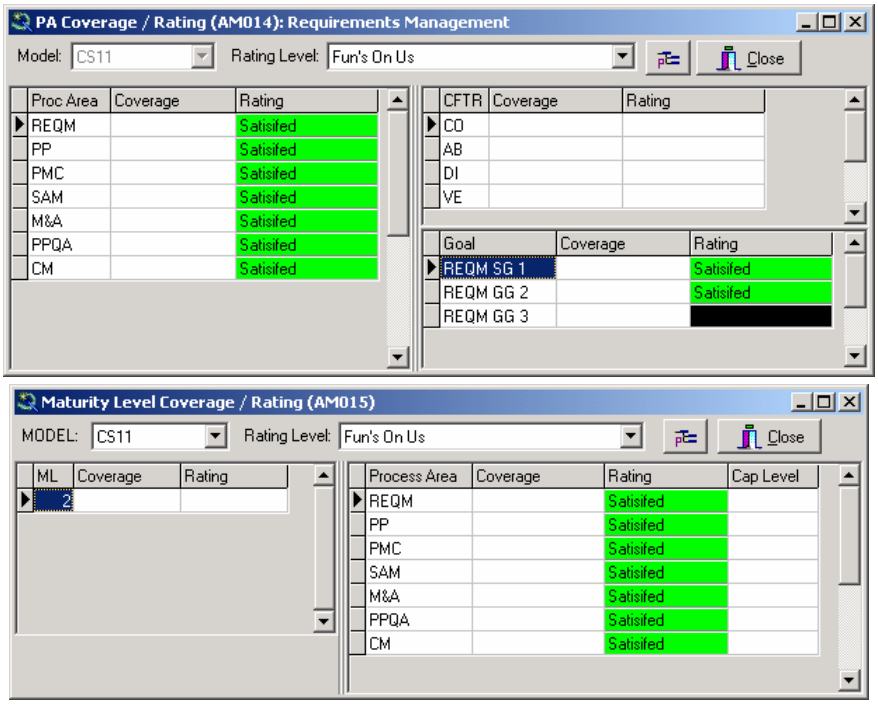

Figura 3.11. Revisión de áreas de proceso y niveles de madurez.

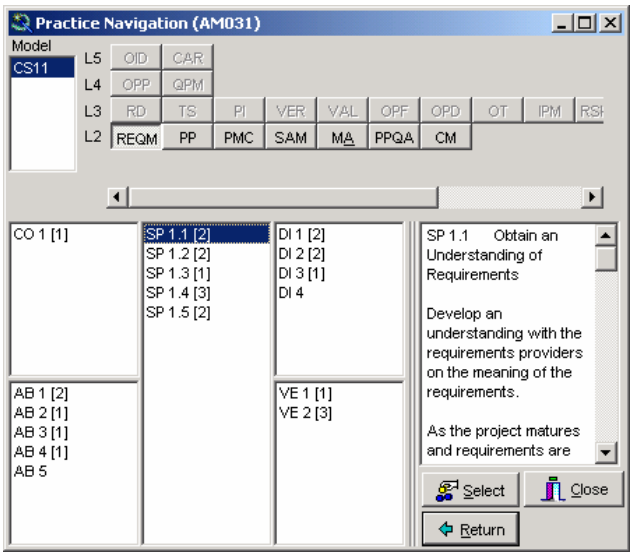

Figura 3.12. Navegador de prácticas.

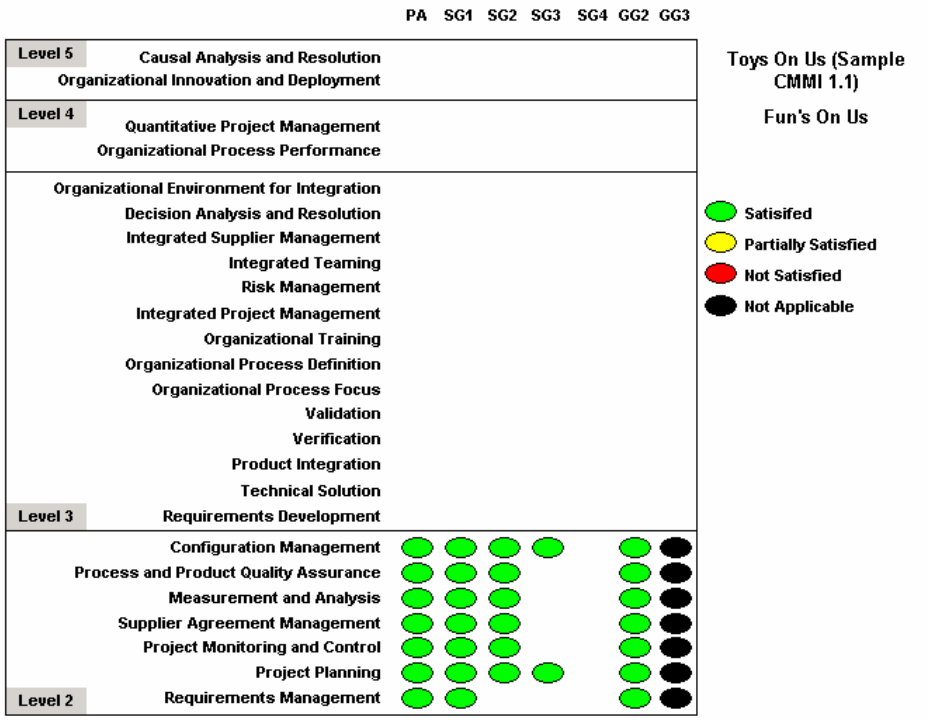

Figura 3.13. Ejemplo de reporte gráfico.

Por último, cabe mencionar los requerimientos de software y hardware de la herramienta:

- Sistema operativo: Windows 95, 98, ME, NT, 2000, XP
- Procesador: mínimo 486 de 33 MHz

- RAM: 32 MB
- Espacio en disco: 25 MB
- Software de terceras partes: motor de base de datos FireBird (viene junto con la herramienta)

3.4.3 Diagnóstico de la situación actual

En base al análisis efectuado en los apartados anteriores, se pueden extraer las siguientes conclusiones:

- Entre las características favorables de CMM-Quest se puede resaltar su facilidad de uso y amigabilidad con el usuario. Como puntos negativos, carece de soporte para el método SCAMPI y la representación por niveles de CMMI, y su nivel de granularidad en las evaluaciones es demasiado grueso, si llegar a analizar las prácticas de la organización.
- Entre las características positivas de Appraisal Wizard, se puede resaltar el amplio soporte que brinda a los métodos de evaluación y modelos de CMMI, y el nivel de granularidad que posee en las evaluaciones, llegando a analizar las prácticas de la organización. Sin embargo, el amplio soporte de métodos y modelos, y el gran nivel de detalle en las evaluaciones, también pueden considerarse como puntos negativos, ya que aportan a que la herramienta sea de difícil utilización y muy poco amigable para los usuarios novatos.
- Como características destacables de ambas herramientas se pueden resaltar las ayudas online y guías de evaluación, el almacenamiento de las evaluaciones para su posterior edición, y las facilidades de generación de reportes y gráficos.
- Como característica desfavorable de ambas herramientas se puede resaltar la falta de generación de conclusiones intermedias para sugerir al usuario. De acuerdo al método SCAMPI, las valoraciones de los objetivos pueden inferirse en base a las valoraciones de las prácticas, y las valoraciones de las áreas de proceso pueden inferirse en base a las valoraciones de los objetivos. Sin embargo, ninguna de las herramientas genera las valoraciones del método para sugerirlas al usuario.
- Otra característica desfavorable de ambas herramientas es que solo se encuentran disponibles para el sistema operativo Windows (en todas sus variantes). No se encontraron herramientas para otros sistemas operativos.
- Otro aspecto desfavorable de ambas herramientas es que carecen de facilidades de navegación sobre la estructura del modelo CMMI (niveles de madurez o capacidad, áreas de proceso, objetivos, prácticas). Cabe mencionar que Appraisal Wizard posee un navegador, pero el mismo es sumamente rudimentario.

Teniendo en cuenta este panorama, se identifican las siguientes mejoras con respecto a la situación actual:

- Brindar un compromiso entre los niveles de granularidad. Esto significa que la evaluación no sea tan gruesa como en CMM-Quest ni tan fina como en Appraisal Wizard. Sería deseable que el usuario pudiera seleccionar el nivel de granularidad a utilizar, pudiendo llegar hasta las prácticas, o solamente hasta los objetivos, o a las áreas de proceso.
- Basar la evaluación en el método SCAMPI y en uno de los modelos de CMMI, de manera de evitar la generalidad que complica el uso de la herramienta.
- Brindar facilidades de navegación sobre la estructura del modelo CMMI, de manera amigable para los usuarios.
- Brindar soporte a otros sistemas operativos (Macintosh, Unixes como Linux, FreeBSD, HP UX).
- Sugerir valoraciones intermedias al usuario, inferidas en base a las valoraciones efectuadas en niveles inferiores.

3.5 Valoración de alternativas y selección de solución

A continuación se sintetizan los resultados del análisis efectuado en el apartado anterior, de manera de poder comparar las dos alternativas contempladas para el análisis de viabilidad: Utilización de herramientas existentes (CMM-Quest o Appraisal Wizard), o desarrollo de nueva herramienta en base a las necesidades.

La tabla 3.1 sintetiza las características de las diferentes alternativas estudiadas.

	CMM-Quest	***Appraisal Wizard***	***Nueva herramienta***
Interfaz de usuario	Fácil, muy amigable	Difícil, poco amigable	Fácil, muy amigable
Tipo de usuario	Novato	Experto	Novato o Experto
Modelos soportados	CMMI-SE/SW (representación Continua)	Gran parte de los CMM y todos los CMMI (ambas representaciones)	CMMI-SW (representación por niveles)
Método SCAMPI	No	Sí	Sí
Nivel de granularidad	Grueso (sólo objetivos)	Fino (hasta prácticas específicas)	Fino o Grueso (seleccionable)
Ayudas online	Sí	Sí	Sí
Navegación de la estructura del modelo	No	No	Sí
Generación de valores sugeridos	No	No	Sí
Selección del nivel de granularidad para la evaluación	No	No	Sí
Portable a múltiples sistemas operativos	No	No	Sí

Tabla 3.1. Comparación de características de las alternativas estudiadas

Analizando la tabla se puede observar que la alternativa de desarrollar una nueva herramienta es la que mejor cubre las necesidades del usuario mencionadas en el apartado 3.1.1 (Necesidades del usuario).

Teniendo además en cuenta las restricciones contempladas en el apartado 3.1.2 (Restricciones), donde se determinó que no existían inconvenientes de carácter económico, técnico o legal, se infiere que la alternativa más conveniente es la de desarrollar una nueva herramienta especialmente enfocada en cubrir las necesidades del usuario. Las razones que justifican esta decisión son las siguientes:

- La nueva herramienta cubre todas las necesidades del usuario.
- Cubre los principales baches y debilidades encontradas en las demás herramientas.
- Su construcción es de bajo costo debido a que puede desarrollarse completamente con herramientas OpenSource.

En los siguientes capítulos del presente trabajo se documentan los diferentes modelos generados durante la construcción de la herramienta (Requerimientos, Análisis, Diseño, Construcción y Pruebas).

4. ESPECIFICACIÓN DE REQUERIMIENTOS DEL SOFTWARE

En este capítulo se presentan los requisitos identificados durante el proceso EVS: Estudio de Viabilidad del Sistema. Los mismos se documentan de acuerdo al formato especificado en el estándar IEEE830-1993 [IEEE 830, 19938], comúnmente conocido como "Especificación de Requerimientos del Software".

4.1 Introducción

4.1.1 Objetivos

El propósito del sistema es asistir a los Ingenieros de Software en la evaluación de una organización o proyecto con respecto al modelo CMMI-SW, siguiendo las reglas establecidas en el método SCAMPI.

4.1.2 Alcance

Esta especificación abarca solamente al sistema denominado "Asistente para la Evaluación de CMMI-SW". No incluye ningún otro sistema existente ni relacionado con este.

4.1.3 Definiciones

Nivel de madurez: el nivel de madurez de una organización provee una manera de predecir el desempeño futuro de una organización en una disciplina específica o en un conjunto de disciplinas. Cada nivel de madurez consiste de un conjunto predefinido de áreas de proceso. Para la evaluación del nivel de madurez alcanzado por una organización, se evalúa la satisfacción de las áreas de proceso que conforman el nivel analizado.

Área de proceso: un área de proceso agrupa un conjunto de prácticas relacionadas que cuando son implementadas de manera colectiva, satisfacen un conjunto de objetivos considerados importantes para el aporte de mejoras significativas en esa área.

Objetivos: los objetivos son las metas deben alcanzarse para satisfacer las áreas de proceso. Se utilizan en las evaluaciones para determinar si un área de proceso es satisfecha o no.

Prácticas: son las actividades llevadas a cabo en una organización o proyecto, que permiten determinar la satisfacción de los objetivos. Se utilizan en las evaluaciones para determinar si un objetivo es satisfecho o no.

4.2 Descripción general

4.2.1 Funciones del producto

El sistema proveerá las siguientes funciones:

- Asistencia en la evaluación de organizaciones o proyectos: el sistema permitirá que el usuario evalúe la adherencia de una organización o proyecto con las prácticas, objetivos, áreas de proceso y niveles de madurez del modelo CMMI-SW. Para asistirlo en su tarea, brindará al usuario guías online y sugerencias en las distintas instancias de evaluación.
- Reporte de resultados: el sistema permitirá que el usuario obtenga un reporte con los resultados de las evaluaciones efectuadas.
- Administración de las evaluaciones: el sistema brindará al usuario la posibilidad de almacenar en archivos las evaluaciones efectuadas. Asimismo, permitirá que el usuario abra los archivos de evaluaciones almacenados para realizar modificaciones en la evaluación.

4.2.2 Características de los usuarios

Los usuarios del sistema son básicamente todos aquellos Ingenieros de Software interesados en la evaluación de una organización o proyecto de acuerdo al modelo CMMI-SW. Dentro de ellos existen al menos los siguientes grupos:

- Usuarios expertos: utilizan el sistema como herramienta para almacenar las evaluaciones de las distintas instancias (prácticas, objetivos, áreas de proceso, nivel de madurez) durante una evaluación formal de acuerdo al método SCAMPI.
- Usuarios no expertos: utilizan el sistema como un asistente para la autoevaluación, siguiendo paso a paso las guías que el mismo provee, de manera de obtener un resultado aproximado que le permita conocer el nivel de madurez de la organización o proyecto.

Desde el punto de vista del sistema no se distinguen los usuarios por su nivel de experiencia, existiendo un único usuario. Las tareas llevadas a cabo por este único usuario abarcan toda la funcionalidad detallada en los requisitos.

4.2.3 Restricciones generales

La utilización del sistema está restringida a un único usuario por vez. No existe la necesidad de atender a usuarios concurrentemente.

4.3 Requisitos específicos

A continuación se muestra la lista de requerimientos específicos del sistema. Los mismos se dividen en "Requisitos funcionales" y "Requisitos no funcionales". Todos los requisitos han sido formalizados en la herramienta *Enterprise Architect* [EA, 2004], de manera de asegurar su trazabilidad con los elementos subsiguientes de análisis y diseño.

4.3.1 Requisitos funcionales

La figura 4.1 muestra gráficamente los requisitos funcionales del sistema, agrupados de acuerdo a las funciones del producto presentadas en el apartado 4.2 (Descripción general).

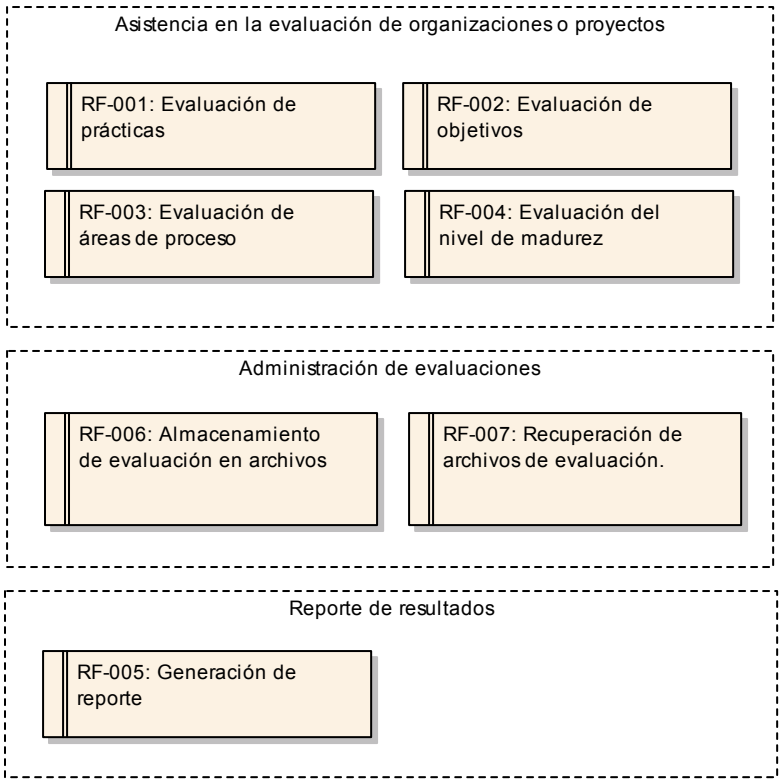

Figura 4.1. Requisitos funcionales.

A continuación se detallan cada una de las funciones y sus requisitos asociados.

Asistencia en la evaluación de organizaciones o proyectos

El sistema permitirá que el usuario evalúe la adherencia de una organización o proyecto con las prácticas, objetivos, áreas de proceso y niveles de madurez del modelo CMMI-SW. Para asistirlo en su tarea, brindará al usuario guías online y sugerencias en las distintas instancias de evaluación.

RF-001: Evaluación de prácticas

Para cada una de las prácticas asociadas a los objetivos, el sistema deberá permitir que el usuario evalúe el nivel de implementación de la misma.

El usuario deberá indicar la presencia de artefactos directos o indirectos que confirmen la implementación de la práctica, y deberá indicar si hay debilidades identificadas. Cada una de las debilidades deberá tener un comentario asociado.

De acuerdo a la información ingresada por el usuario, el sistema asignará alguno de los siguientes valores a la práctica de acuerdo al método SCAMPI:

- CI: completamente implementada
- AI: ampliamente implementada
- PI: parcialmente implementada

- NI: no implementada

Las prácticas podrán evaluarse a nivel de proyecto o a nivel de organización. En caso de evaluarse a nivel de proyecto para más de un proyecto, el sistema asignará los valores a nivel de organización de acuerdo a la combinación de los valores a nivel de proyecto siguiendo las reglas del método SCAMPI (ver reglas en el apartado 6.2 “Modelo del negocio” del capítulo 6).

RF-002: Evaluación de objetivos

Para cada uno de los objetivos asociados a las áreas de proceso, el sistema deberá permitir que el usuario evalúe la satisfacción del mismo. El usuario podrá evaluar los objetivos de dos maneras distintas:

1) Evaluando primero las prácticas asociadas al objetivo, de acuerdo a lo especificado en el requisito RF-001. En esta variante, el sistema asignará los valores "Satisfecho" o "No satisfecho" de acuerdo a la combinación de los valores asignados a las prácticas (ver reglas en el apartado 6.2 “Modelo del negocio” del capítulo 6).
2) Asignando directamente los valores "Satisfecho" o "No satisfecho", con un comentario que justifique su valoración.

RF-003: Evaluación de áreas de proceso

Para cada una de las áreas de proceso asociadas a los niveles de madurez, el sistema deberá permitir que el usuario evalúe la satisfacción o insatisfacción de la misma. El usuario podrá evaluar las áreas de proceso de dos maneras distintas:

1) Evaluando primero los objetivos asociados al área de proceso, de acuerdo a lo especificado en el requisito RF-002. En esta variante, el sistema asignará los valores "Satisfecha" o "No satisfecha" de acuerdo a la combinación de los valores asignados a los objetivos (ver reglas en el apartado 6.2 “Modelo del negocio” del capítulo 6).
2) Asignando directamente los valores "Satisfecha" o "No satisfecha", con un comentario que justifique su valoración.

RF-004: Evaluación del nivel de madurez

Para cada uno de los niveles de madurez, el sistema deberá permitir que el usuario evalúe si el mismo es alcanzado o no.

El usuario podrá seleccionar el nivel de madurez a evaluar, y el sistema indicará al usuario las áreas de proceso asociadas al nivel de madurez seleccionado, de manera que el usuario las evalúe de acuerdo a lo especificado en el requisito RF-003.

Administración de evaluaciones

El sistema brindará al usuario la posibilidad de almacenar en archivos las evaluaciones efectuadas. Asimismo, permitirá que el usuario abra los archivos de evaluaciones almacenados para realizar modificaciones en la evaluación.

RF-006: Almacenamiento de evaluación en archivos

El sistema deberá permitir que el usuario almacene en un archivo los resultados de una evaluación completa o en curso. Para ello, brindará una interfaz donde el usuario podrá indicar el archivo donde se almacenará la evaluación.

RF-007: Recuperación de archivos de evaluación.

El sistema deberá permitir que el usuario abra una evaluación guardada en archivo para su modificación.

Reporte de resultados

El sistema permitirá obtener reportes con los resultados de las evaluaciones efectuadas.

RF-005: Generación de reporte

El sistema deberá permitir que el usuario genere un reporte con los resultados de una evaluación completa.

El reporte mostrará las valoraciones sugeridas por el sistema y las asignadas por el usuario para cada uno de los componentes del modelo (Prácticas, Objetivos, Áreas de proceso, Niveles de madurez).

4.3.2 Requisitos no funcionales

La figura 4.2 muestra gráficamente los requisitos no funcionales del sistema, agrupados de acuerdo a características de Usabilidad y de Portabilidad. No se definieron requisitos para otros aspectos no funcionales.

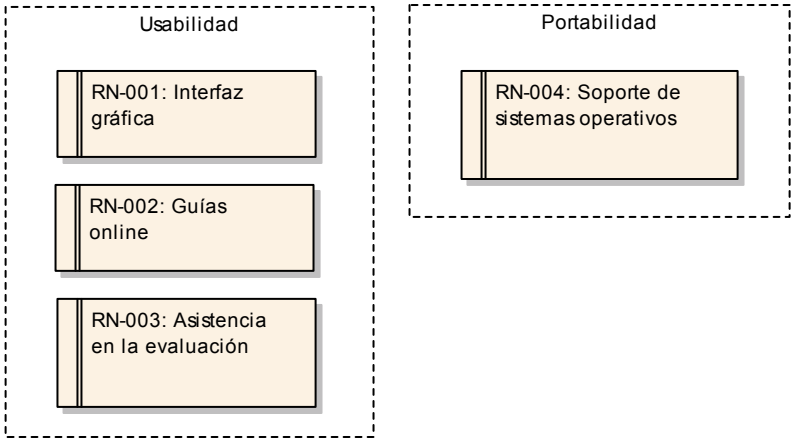

Figura 4.2. Requisitos no funcionales.

Usabilidad

Requerimientos no funcionales de usabilidad del sistema.

RN-001: Interfaz gráfica

El sistema deberá proveer al usuario una interfaz gráfica interactiva que muestre la estructura general del modelo CMMI-SW. El usuario podrá navegar sobre los distintos componentes del modelo para realizar las evaluaciones.

La interfaz deberá contar además con menúes y ventanas que faciliten las tareas de evaluación.

RN-002: Guías online

El sistema deberá proveer al usuario guías online que contengan la documentación completa del modelo CMMI-SW. El usuario podrá consultar estas guías para efectuar las tareas de evaluación.

RN-003: Asistencia en la evaluación

El sistema deberá asistir al usuario en la valorización de las distintas instancias de evaluación. Para cada instancia, brindará sugerencias aplicando las reglas del método SCAMPI sobre las valorizaciones existentes.

Ej: para la evaluación de los objetivos, tomará los valores asignados a las prácticas del mismo, sugiriendo al usuario un valor apropiado.

El usuario podrá ignorar las sugerencias del sistema y asignar los valores que desee, con un comentario justificativo.

Portabilidad

Requerimientos no funcionales de portabilidad del sistema.

RN-004: Soporte de sistemas operativos

El sistema deberá poder ejecutarse sobre varios sistemas operativos, entre los cuales se incluyen Windows, MacOS, Unixes como Linux, FreeBSD, HP UX.

4.4 Identificación de actores y casos de uso

Luego de definidos los requisitos del sistema, se identificaron los actores y casos de uso a partir de los mismos. Para cada actor y caso de uso identificado, se elaboró una breve descripción que brinda una idea inicial de la complejidad del mismo.

A continuación se muestra un reporte generado con la herramienta *Enterprise Architect* mostrando los actores y casos de uso identificados.

Reporte: Modelo de casos de uso

El diagrama de la figura 4.3 muestra los actores y casos de uso del sistema.

Usuario

public Actor: Representa al usuario (experto o novato) que utiliza el sistema como herramienta de asistencia para la evaluación de una organización o proyecto de software.

Administrar evaluaciones

public Use Case: Brinda la posibilidad de iniciar evaluaciones, de almacenar en un archivo los resultados de una evaluación completa o en curso, y de recuperar una evaluación almacenada en archivo para su modificación.

Para la selección de los archivos, el sistema provee una interfaz gráfica que permite navegar en el sistema de archivos.

Administrar observaciones

public Use Case: Brinda la posibilidad de crear nuevas observaciones, y de modificar o eliminar observaciones existentes. El usuario podrá vincular y desvincular estas observaciones con las prácticas del modelo.

Cada observación tiene un atributo que la caracteriza como debilidad o fortaleza y otro que indica si su impacto es significativo o no.

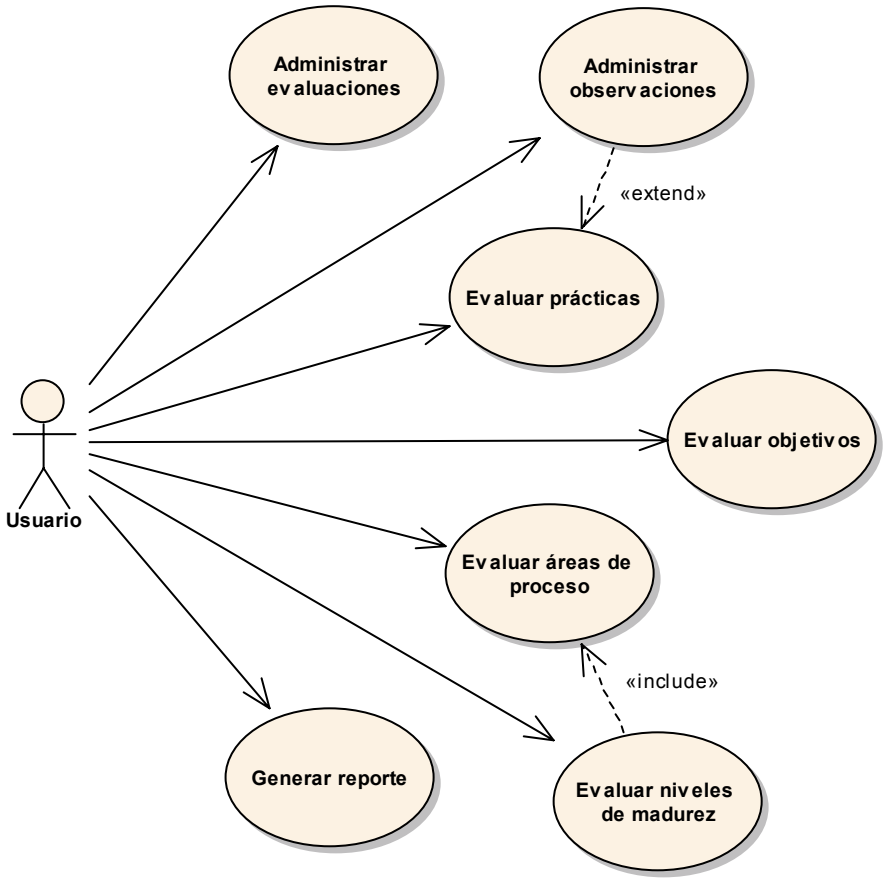

Figura 4.3. Actores y Casos de uso del sistema.

Evaluar niveles de madurez

public Use Case: Brinda la posibilidad de evaluar el nivel de madurez del modelo CMMI-SW alcanzado.

El usuario selecciona el nivel de madurez que desea evaluar y el sistema le indica las áreas de proceso asociadas. El usuario las evalúa utilizando el caso de uso Evaluar áreas de proceso y finalmente el sistema le responde si el nivel es alcanzado o no.

El usuario puede consultar en las guías online del sistema toda la información relativa al nivel de madurez que está analizando (criterios para su evaluación, áreas de proceso asociadas, etc).

Evaluar áreas de proceso

public Use Case: Brinda la posibilidad de evaluar el nivel de satisfacción de cada una de las áreas de proceso asociadas a los niveles de madurez del modelo CMMI-SW.

Para cada área de proceso, el usuario puede asignar alguno de los valores "Satisfecho" o "No satisfecho", con un comentario que justifique su valoración.

Si el usuario utilizó previamente el caso de uso Evaluar objetivos para evaluar los objetivos del área de proceso bajo análisis, el sistema sugerirá los valores "Satisfecho" o "No satisfecho" de acuerdo a la combinación de los valores asignados a las objetivos.

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

El usuario puede consultar en las guías online del sistema toda la información relativa al área de proceso que está analizando (criterios para su evaluación, objetivos asociados, etc).

Evaluar objetivos

public Use Case: Brinda la posibilidad de evaluar el nivel de satisfacción de cada uno de los objetivos asociados a las áreas de proceso del modelo CMMI-SW.

Para cada objetivo, el usuario puede asignar alguno de los valores "Satisfecho" o "No satisfecho", con un comentario que justifique su valoración.

Si el usuario utilizó previamente el caso de uso Evaluar prácticas para evaluar las prácticas asociadas al objetivo bajo análisis, el sistema sugerirá los valores "Satisfecho" o "No satisfecho" de acuerdo a la combinación de los valores asignados a las prácticas.

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

El usuario puede consultar en las guías online del sistema toda la información relativa al objetivo que está analizando (criterios para su evaluación, prácticas asociadas, etc).

Evaluar prácticas

public Use Case: Brinda la posibilidad de evaluar el nivel de implementación de cada una de las prácticas asociadas a los objetivos del modelo CMMI-SW.

El usuario indica la presencia de artefactos directos o indirectos que confirmen la implementación de la práctica, e indica si hay observaciones caracterizadas como debilidad asociadas con la práctica.

De acuerdo a la información ingresada por el usuario, el sistema sugerirá alguno de los siguientes valores:

- CI: completamente implementada
- AI: ampliamente implementada
- PI: parcialmente implementada
- NI: no implementada

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

Las prácticas podrán evaluarse a nivel de proyecto o a nivel de organización. En caso de evaluarse a nivel de proyecto para más de un proyecto, el sistema asignará los valores a nivel de organización de acuerdo a la combinación de los valores a nivel de proyecto siguiendo las reglas del método SCAMPI.

El usuario puede consultar en las guías online del sistema toda la información relativa a la práctica que está analizando (artefactos directos e indirectos, criterios para su evaluación, etc).

Generar reporte

public Use Case: Brinda la posibilidad de generar un reporte con los resultados de la evaluación.

El reporte muestra las valoraciones sugeridas, las valoraciones elegidas, y las justificaciones provistas por el usuario, para cada uno de los componentes del modelo (Niveles de madurez, Áreas de proceso, Objetivos, Prácticas).

5. PLAN GENERAL DEL PROYECTO

En este capítulo se documentan los productos de salida de las interfaces "Gestión de proyectos", "Gestión de la configuración", "Aseguramiento de la calidad", y "Seguridad" de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

5.1 Gestión del proyecto

5.1.1 Estimación del esfuerzo

Para la estimación del esfuerzo se aplicará la técnica de Puntos de caso de uso (*UCP, Use Case Points*) [Peralta, M.L. 2004][Ribu, K. 2001]. La técnica toma como entrada los casos de uso identificados para el sistema bajo análisis y produce un estimado en horas hombre del esfuerzo necesario para el desarrollo del mismo.

A continuación de detallan los pasos efectuados para la aplicación de la técnica. Los conceptos serán expuestos a medida que se relata su aplicación.

5.1.1.1 Resumen de actores y casos de uso

Como punto de partida para la aplicación de la técnica es necesario haber efectuado la identificación de actores y casos de uso del sistema. Los actores y casos de

uso identificados se detallaron en el apartado 4.4 (Identificación de actores y casos de uso) del capítulo 4. Tomando como base la información contenida en dicho apartado, se continúa con la aplicación de la técnica.

5.1.1.2 Cálculo de los Puntos de caso de uso sin ajustar

El paso siguiente en la técnica consiste en determinar la complejidad asignada a cada uno de los actores y casos de uso identificados.

Para los actores se sigue el criterio mostrado en la tabla 5.1.

Tipo de Actor	*Descripción*	*Peso*
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3

Tabla 5.1. Criterio para la asignación de complejidad a los actores.

Para los casos de uso, el criterio a seguir es el que se muestra en la tabla 5.2.

Tipo de Caso de Uso	*Descripción*	*Peso*
Simple	El Caso de Uso contiene de 1 a 3 transacciones (*)	5
Medio	El Caso de Uso contiene de 4 a 7 transacciones (*)	10
Complejo	El Caso de Uso contiene más de 8 transacciones (*)	15

Tabla 5.2. Criterio para la asignación de complejidad a los Casos de uso. (*) Una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia. Las transacciones tienen una cierta correspondencia con los flujos dentro del caso de uso.

Una vez que se han asignado los pesos a los actores y a los casos de uso de acuerdo a su dificultad, se calculan los Puntos de caso de uso sin ajustar de acuerdo a la fórmula 5.1.

$$UUCP = UAW + UUCW \qquad \textbf{(Fórmula 5.1)}$$

donde,

***UUCP**:* Puntos de Casos de Uso sin ajustar
***UAW**:* Sumatoria de los factores de peso de los actores
***UUCW**:* Sumatoria de los factores de peso de los casos de uso

La tabla 5.3 muestra los valores asignados para los actores y casos de uso del sistema.

Elemento	*Complejidad*	*Peso*	*Justificación*
Generar reporte	Simple	5	Posee 3 transacciones (una para cada nivel de reporte)
Evaluar niveles de madurez	Simple	5	Posee a lo sumo 2 transacciones (una para cada modo de evaluación)
Evaluar áreas de proceso	Simple	5	Posee una única transacción
Evaluar objetivos	Simple	5	Posee una única transacción
Administrar observaciones	Media	10	Posee 5 transacciones (alta, baja, modificación, vinculación y desvinculación)
Evaluar prácticas	Simple	5	Posee una única transacción
Administrar evaluaciones	Simple	5	Posee 2 transacciones (almacenamiento y recuperación)
Usuario	Complejo	3	Se trata de una persona interactuando con el sistema a través de una interfaz gráfica
Puntos de caso de uso sin ajustar (UUCP)		43	

Tabla 5.3. Valores asignados a los actores y casos de uso del sistema, y valor resultante.

5.1.1.3 Cálculo de los Puntos de caso de uso ajustados

Continuando con la aplicación de la técnica, los puntos de caso de uso calculados en el apartado anterior deben ajustarse de acuerdo a un conjunto de parámetros que miden la complejidad técnica y los factores de ambiente del proyecto.

El coeficiente de incidencia de la complejidad técnica se obtiene asignando valores a los criterios que se muestran en la tabla 5.4. Los valores a asignar van desde 0 hasta 5 de acuerdo a la incidencia del criterio en el proyecto.

Factor	*Descripción*	*Peso*
TCF01	Sistema distribuído	2
TCF02	Objetivos de performance o tiempo de respuesta	1
TCF03	Eficiencia del usuario final	1
TCF04	Procesamiento interno complejo	1
TCF05	El código debe ser reutilizable	1
TCF06	Facilidad de instalación	0.5
TCF07	Facilidad de uso	0.5
TCF08	Portabilidad	2
TCF09	Facilidad de cambio	1
TCF10	Concurrencia	1
TCF 11	Incluye objetivos especiales de seguridad	1
TCF 12	Provee acceso directo a terceras partes	1
TCF 13	Se requieren facilidades especiales de entrenamiento a usuarios	1

Tabla 5.4. Criterios para el coeficiente de incidencia de la complejidad técnica.

Finalmente, el coeficiente de incidencia para la complejidad técnica se calcula de acuerdo a la fórmula 5.2.

$$TCF = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor\ asignado_i) \qquad \textbf{(Fórmula 5.2)}$$

El coeficiente de incidencia de los factores de ambiente se obtiene asignando valores a los criterios que se muestran en la tabla 5.5. Los valores a asignar van desde 0 hasta 5 de acuerdo a la incidencia del criterio en el proyecto.

Factor	*Descripción*	*Peso*
ECF01	Familiaridad con el modelo de proyecto utilizado	1.5
ECF02	Experiencia en la aplicación	0.5
ECF03	Experiencia en orientación a objetos	1
ECF04	Capacidad del analista líder	0.5
ECF05	Motivación	1
ECF06	Estabilidad de los requerimientos	2
ECF07	Personal part-time	-1
ECF08	Dificultad del lenguaje de programación	-1

Tabla 5.5. Criterios para el coeficiente de incidencia de los factores de ambiente.

Finalmente, el coeficiente de incidencia para el factor de ambiente se calcula de acuerdo a la fórmula 5.3.

$$EF = 1.4 - 0.03 \times \Sigma (Peso_i \times Valor\ asignado_i) \qquad \textbf{(Fórmula 5.3)}$$

La tabla 5.6 muestra los valores de complejidad técnica asignados para el sistema.

Factor	*Descripción*	*Justificación*	*Peso*	*Valor*
TCF01	Sistema distribuído	El sistema no es distribuido.	2	0
TCF02	Objetivos de performance o tiempo de respuesta	Los objetivos de performance son fáciles de lograr.	1	1
TCF03	Eficiencia del usuario final	Los requerimientos de eficiencia son altos.	1	5
TCF04	Procesamiento interno complejo	El procesamiento interno es simple.	1	1
TCF05	El código debe ser reutilizable	Sólo dentro del sistema, por buenas prácticas de programación.	1	3
TCF06	Facilidad de instalación	Debe ser muy fácil de instalar.	0.5	5
TCF07	Facilidad de uso	Debe ser muy fácil de utilizar.	0.5	5
TCF08	Portabilidad	El sistema debe ser portable a varios sistemas operativos.	2	5
TCF09	Facilidad de cambio	Debe ser relativamente fácil de modificar.	1	3
TCF10	Concurrencia	No hay requerimientos de concurrencia. El sistema es monousuario.	1	0
TCF11	Incluye objetivos especiales de seguridad	No hacen falta características especiales de seguridad.	1	1
TCF12	Provee acceso directo a terceras partes	No debe proveer acceso a terceros. Sólo al usuario.	1	0
TCF13	Se requieren facilidades especiales de entrenamiento a usuarios	Debe tener ayudas online y guías para el usuario.	1	5
Coeficiente de incidencia de la complejidad técnica (TCF)				0.89

Tabla 5.6. Valores asignados a los factores de complejidad técnica y coeficiente obtenido.

La tabla 5.7 muestra los valores de factores de ambiente asignados para el sistema.

Factor	*Descripción*	*Justificación*	*Peso*	*Valor*
ECF01	Familiaridad con el modelo de proyecto utilizado	Se conocen los conceptos de Métrica 3 pero es la primera vez que se aplica.	1.5	3
ECF02	Experiencia en la aplicación	Se tiene gran experiencia en Java, aunque no en aplicaciones standalone.	0.5	2
ECF03	Experiencia en orientación a objetos	Se tiene gran experiencia en orientación a objetos.	1	4
ECF04	Capacidad del analista líder	El analista tiene mucha experiencia en casos de uso.	0.5	4
ECF05	Motivación	Se tiene una alta motivación para completar el proyecto.	1	5
ECF06	Estabilidad de los requerimientos	Los requerimientos son altamente estables.	2	4
ECF07	Personal part-time	Todos los involucrados en el proyecto son part-time.	-1	5
ECF08	Dificultad del lenguaje de programación	El lenguaje a utilizar es de dificultad media.	-1	3
Coeficiente de incidencia del factor de ambiente (ECF)				0.935

Tabla 5.7. Valores asignados a los factores de ambiente y coeficiente obtenido.

Finalmente, los Puntos de caso de uso ajustados se obtienen mediante la fórmula 5.4.

$$UCP = UUCP \times TCF \times EF \qquad \textbf{(Fórmula 5.4)}$$

donde,

***UCP**:* Puntos de Casos de Uso ajustados
***UUCP**:* Puntos de Casos de Uso sin ajustar
***TCF**:* Coeficiente de incidencia de la complejidad técnica
***ECF**:* Coeficiente de incidencia del factor ambiente

Puntos de caso de uso ajustados (UCP)	35

5.1.1.4 Cálculo del esfuerzo

El esfuerzo se calcula multiplicando la cantidad de puntos de caso de uso ajustados por un factor de conversión que lo convierte en horas/hombre. Inicialmente se sugirió para este coeficiente un valor de 20 horas/hombre por punto de caso de uso, pero con posterioridad surgieron al menos tres variantes [Ribu, K. 2001]:

- La primera de ellas sugiere la siguiente aproximación:
  - Se contabilizan entre los factores ECF01 a ECF06 cuántos están por debajo del valor medio (3).

  - Se contabilizan entre los factores ECF07 a ECF08 cuántos están por encima del valor medio (3).
  - Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/punto de caso de uso, es decir, un punto de caso de uso toma 20 horas-hombre.
  - Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/ punto de caso de uso, es decir, un punto de caso de uso toma 28 horas-hombre.
  - Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.
- La segunda se basa en estadísticas de la industria del software y plantea que el valor del coeficiente varía entre 15 y 30 horas-hombre/punto de caso de uso, de acuerdo a las características del proyecto
- La tercera sugiere un valor más conservador de 36 horas-hombre/punto de caso de uso.

Para nuestro caso se tiene que entre ECF01 y ECF06 hay un solo valor por debajo de 3 (ECF02), y entre ECF07 y ECF08 hay un solo valor por encima de 3 (ECF07).

Con estos valores y adoptando la primera de las aproximaciones mencionadas, el factor de conversión resulta de 20 horas-hombre/punto de caso de uso.

Esfuerzo estimado	700 horas/hombre

5.1.2 Planificación

En esta sección se presenta la planificación general del proyecto, partiendo de una estrategia para el desarrollo y definiendo posteriormente el calendario completo donde se incluyen las distintas actividades a realizar.

5.1.2.1 Estrategia de desarrollo

Para el desarrollo del sistema se selecciona un ciclo de vida basado en las fases del Proceso Unificado de Desarrollo [Booch & Jacobson & Rumbaugh, 2000]. De acuerdo a este proceso, las fases del desarrollo son las siguientes:

- Incepción: orientada básicamente a analizar la viabilidad del software, en esta fase se determina el alcance y se estima el esfuerzo necesario para el desarrollo.

- Elaboración: orientada a estabilizar la arquitectura, en esta fase se desarrollan y se prueban los casos de uso más significativos desde el punto de vista arquitectónico, con el fin de fijar la estructura de funcionamiento del sistema.
- Construcción: orientada a obtener la versión beta del sistema, en esta fase se completa y se prueba el resto de la funcionalidad del mismo.
- Transición: orientada a obtener la versión estable del sistema, en esta fase se completa el pasaje a producción del mismo.

En cada una de estas fases se llevan a cabo iteraciones, cada una de las cuales comprende actividades de Análisis, Diseño, Construcción y Pruebas. Por este motivo, el ciclo de vida se denomina Iterativo.

Para el caso del sistema a desarrollar, se plantean 4 iteraciones, cada una de ellas ligadas a una fase del Proceso Unificado. No es necesario plantear más de una iteración por fase, debido a que no existen riesgos significativos que mitigar. Esto hace que en una única iteración se puedan cumplir los objetivos de la fase.

Como puede observarse en las descripciones de las iteraciones, las mismas contemplan actividades correspondientes a varios procesos de Métrica V3.

- **Iteración 1 – Viabilidad** (corresponde a la fase de Incepción)
  Los objetivos principales de esta iteración son los siguientes:

  - Analizar los sistemas existentes, describir posibles mejoras, y establecer los requerimientos de alto nivel del nuevo sistema.
  - Estimar el esfuerzo necesario para el desarrollo del sistema y establecer el plan general del proyecto, incluyendo planes de Gestión de configuración, Calidad y Seguridad.

  Dentro de esta iteración se contemplan las actividades del proceso EVS con las interfaces que lo afectan (Gestión de la configuración, Aseguramiento de la calidad y Seguridad), y las actividades de inicio del proyecto de la interfaz Gestión de proyectos.

- **Iteración 2 – Arquitectura** (corresponde a la fase de Elaboración)
  Los objetivos principales de esta iteración son los siguientes:

  - Definir la arquitectura del sistema, contemplando sus mecanismos principales de funcionamiento.
  - Construir un prototipo operativo que contemple los siguientes elementos:
    - las interfaces gráficas de usuario con navegación del modelo y guías online

- el soporte a múltiples sistemas operativos
- el almacenamiento y recuperación de archivos
- la asistencia en la evaluación de prácticas, objetivos y áreas de proceso

El alcance de las facilidades de evaluación se limitará a un área de proceso, y uno o dos de sus objetivos con las prácticas asociadas al mismo.

Dentro de esta iteración se contemplan las actividades de los procesos ASI, DSI, CSI e IAS, junto con las interfaces que los afectan (Gestión de la configuración, Aseguramiento de la calidad, Seguridad y Gestión de proyectos).

- **Iteración 3 – Construcción** (corresponde a la fase de Construcción)
  Los objetivos principales de esta iteración son los siguientes:

  - Completar la funcionalidad del prototipo construido en la iteración anterior, de manera de cubrir los requerimientos restantes del sistema. Esto incluye:
    - la generación de reportes
    - la evaluación de todos los niveles de madurez, áreas de proceso, objetivos y prácticas
  - Alcanzar la Versión 1.0 beta de la herramienta.

  Dentro de esta iteración se contemplan las actividades de los procesos ASI, DSI, CSI e IAS, junto con las interfaces que los afectan (Gestión de la configuración, Aseguramiento de la calidad, Seguridad y Gestión de proyectos).

- **Iteración 4 – Cierre** (corresponde a la fase de Transición)
  Los objetivos principales de esta iteración son los siguientes:

  - Completar la documentación de usuario del sistema.
  - Completar las pruebas de aceptación del sistema.
  - Efectuar las correcciones necesarias que surjan como resultado de las pruebas.
  - Obtener la Versión 1.0 estable de la herramienta.

  Dentro de esta iteración se contemplan las actividades de los procesos CSI e IAS, junto con las interfaces que lo afectan (Gestión de la configuración, Aseguramiento de la calidad, Seguridad y Gestión de proyectos).

5.1.2.2 Estructura de actividades

A continuación se muestran las actividades y tareas de Métrica v3 [Métrica V3, 2000] que se llevarán a cabo durante el desarrollo de la tesis. Para cada proceso o interfaz de la metodología, se construye una tabla donde se muestran las actividades y tareas a llevar a cabo, y los productos de salida de cada una de ellas.

Proceso PSI: Planificación de Sistemas de Información
El sistema propuesto para la tesis es básicamente un sistema independiente que no se sitúa en ningún ámbito organizativo determinado. De esta manera, la necesidad no es la de elaborar un Plan de Sistemas de Información para una organización en particular, sino de desarrollar un sistema que eventualmente puede ser utilizado en cualquier ámbito organizativo.

Tabla 5.8. Actividades, tareas y productos del proceso PSI.

Proceso EVS: Estudio de Viabilidad del Sistema		
Actividades	**Tareas**	**Productos**
EVS1 Establecimiento del alcance del sistema	EVS1.1 Estudio de la solicitud	Descripción general del sistema Catálogo de objetivos del EVS Catálogo de requisitos
	EVS1.2 Identificación del alcance del sistema	Descripción general del sistema Catálogo de requisitos Catálogo de usuarios
	EVS1.3 Especificación del alcance del EVS	Catálogo de objetivos del EVS Catálogo de usuarios Plan de trabajo
EVS2 Estudio de la situación actual	EVS2.1 Valoración del estudio de la situación actual	Descripción de la situación actual
	EVS2.3 Descripción de los sistemas de información existentes	Descripción de la situación actual
	EVS2.4 Realización del diagnóstico de la situación actual	Diagnóstico de la situación actual
EVS3 Definición de requisitos del sistema	EVS3.2 Identificación de requisitos	Identificación de requisitos
	EVS3.3 Catalogación de requisitos	Catálogo de requisitos
EVS4 Estudio de alternativas de solución	EVS4.1 Preselección de alternativas de solución	Descripción de la situación actual
	EVS4.2 Descripción de alternativas de solución	Restricciones Diagnóstico de la situación actual
EVS6 Selección de la solución	EVS6.2 Evaluación de las alternativas y selección	Valoración de alternativas y selección de la solución
	EVS6.3 Aprobación de la solución	Valoración de alternativas y selección de la solución

Tabla 5.9. Actividades, tareas y productos del proceso EVS.

Proceso ASI: Análisis del Sistema de Información		
Actividades	**Tareas**	**Productos**
ASI1 Definición del sistema	ASI1.1 Determinación del alcance del sistema	Catálogo de requisitos Glosario Modelo de negocio Modelo de dominio
	ASI1.2 Identificación del entorno tecnológico	Catálogo de requisitos Descripción general del entorno tecnológico
	ASI1.4 Identificación de usuarios participantes y finales	Catálogo de usuarios Planificación
ASI2 Establecimiento de requisitos	ASI2.1 Obtención de requisitos	Catálogo de requisitos Modelo de casos de uso
	ASI2.2 Especificación de casos de uso	Catálogo de requisitos Modelo de casos de uso Especificación de casos de uso
	ASI2.3 Análisis de requisitos	Catálogo de requisitos Modelo de casos de uso Especificación de casos de uso
	ASI2.4 Validación de requisitos	Catálogo de requisitos Modelo de casos de uso Especificación de casos de uso
ASI3 Identificación de subsistemas de análisis	ASI3.1 Determinación de subsistemas de análisis	Descripción de subsistemas de análisis Descripción de interfaces entre subsistemas
	ASI3.2 Integración de subsistemas de análisis	Desarrollo y aceptación Descripción de subsistemas de análisis Descripción de interfaces entre subsistemas
ASI4 Análisis de los casos de uso	ASI4.1 Identificación de clases asociadas a un caso de uso	Modelo de clases de análisis
	ASI4.2 Descripción de la interacción entre objetos	Análisis de la realización de los casos de uso
ASI5 Análisis de las clases	ASI5.1 Identificación de responsabilidades y atributos	Modelo de clases de análisis Comportamiento de clases de análisis
	ASI5.2 Identificación de asociaciones y agregaciones	Modelo de clases de análisis
	ASI5.3 Identificación de generalizaciones	Modelo de clases de análisis
ASI8 Definición de interfaces de usuario	ASI8.1 Especificación de principios generales de la interfaz	Principios generales de la interfaz
	ASI8.4 Especificación del comportamiento dinámico de la interfaz	Modelo de navegación de interfaz de pantalla Prototipo de interfaz interactiva
	ASI8.5 Especificación de formatos de impresión	Formatos de impresión Prototipo de interfaz de impresión

Proceso ASI: Análisis del Sistema de Información		
Actividades	**Tareas**	**Productos**
ASI9 Análisis de consistencia y especificación de requisitos	ASI9.1 Verificación de los modelos	Especificación de interfaz de usuario Modelo de casos de uso Especificación de casos de uso Descripción de los subsistemas de análisis Descripción de interfaces entre subsistemas Modelo de clases de análisis Comportamiento de clases de análisis Análisis de la realización de los casos de uso
	ASI9.2 Análisis de consistencia entre modelos	Resultado de análisis de consistencia Especificación de interfaz de usuario Modelo de casos de uso Especificación de casos de uso Descripción de los subsistemas de análisis Descripción de interfaces entre subsistemas Modelo de clases de análisis Comportamiento de clases de análisis Análisis de la realización de los casos de uso
	ASI9.3 Validación de los modelos	Especificación de interfaz de usuario Modelo de casos de uso Especificación de casos de uso Descripción de los subsistemas de análisis Descripción de interfaces entre subsistemas Modelo de clases de análisis Comportamiento de clases de análisis Análisis de la realización de los casos de uso
ASI10 Especificación del plan de pruebas	ASI10.1 Definición del alcance de las pruebas	Plan de pruebas
	ASI10.2 Definición de requisitos del entorno de pruebas	Plan de pruebas
	ASI10.3 Definición de las pruebas de aceptación del sistema	Plan de pruebas

Tabla 5.10. Actividades, tareas y productos del proceso ASI.

Proceso DSI: Diseño del Sistema de Información		
Actividades	**Tareas**	**Productos**
DSI1 Definición de la arquitectura del sistema	DSI1.1 Definición de niveles de arquitectura	Diseño de la arquitectura del sistema
	DSI1.2 Identificación de requisitos de diseño y construcción	Catálogo de requisitos
	DSI1.5 Identificación de subsistemas de diseño	Diseño de la arquitectura del sistema
	DSI1.6 Especificación del entorno tecnológico	Entorno tecnológico del sistema
DSI2 Diseño de la arquitectura de soporte	DSI2.1 Diseño de subsistemas de soporte	Diseño detallado de subsistemas de soporte
	DSI2.2 Identificación de mecanismos genéricos de diseño	Mecanismos genéricos de diseño y construcción
DSI3 Diseño de casos de uso reales	DSI3.1 Identificación de clases asociadas a un caso de uso	Diseño de la realización de los casos de uso
	DSI3.2 Diseño de la realización de los casos de uso	Diseño de la realización de los casos de uso
	DSI3.3 Revisión de la interfaz de usuario	Diseño de interfaz de usuario
	DSI3.4 Revisión de subsistemas de diseño e interfaces	Diseño de la realización de los casos de uso
DSI4 Diseño de clases	DSI4.1 Identificación de clases adicionales	Modelo de clases de diseño
	DSI4.2 Diseño de asociaciones y agregaciones	Modelo de clases de diseño
	DSI4.3 Identificación de atributos de las clases	Modelo de clases de diseño
	DSI4.4 Identificación de operaciones de las clases	Modelo de clases de diseño Comportamiento de clases de diseño
	DSI4.5 Diseño de la jerarquía	Modelo de clases de diseño
DSI7 Verificación y aceptación de la arquitectura del sistema	DSI7.1 Verificación de las especificaciones de diseño	Entorno tecnológico del sistema Diseño de la arquitectura del sistema Diseño detallado de subsistemas de soporte Diseño de interfaz de usuario Diseño de la realización de los casos de uso Modelo de clases de diseño Comportamiento de clases de diseño
	DSI7.2 Análisis de consistencia de las especificaciones de diseño	Entorno tecnológico del sistema Diseño de la arquitectura del sistema Diseño detallado de subsistemas de soporte Diseño de interfaz de usuario Diseño de la realización de los casos de uso Modelo de clases de diseño Comportamiento de clases de diseño

Proceso DSI: Diseño del Sistema de Información		
Actividades	**Tareas**	**Productos**
DSI8 Generación de especificaciones de construcción	DSI8.1 Especificación del entorno de construcción	Especificaciones de construcción del sistema de información
	DSI8.2 Definición de componentes y subsistemas de construcción	Especificaciones de construcción del sistema de información
DSI10 Especificación técnica del plan de pruebas	DSI10.1 Especificación del entorno de pruebas	Plan de pruebas
	DSI10.2 Especificación técnica de niveles de prueba	Plan de pruebas
	DSI10.3 Revisión de la planificación de pruebas	Plan de pruebas
DSI11 Establecimiento de requisitos de implantación	DSI11.1 Especificación de requisitos de documentación de usuario	Catálogo de requisitos
	DSI11.2 Especificación de requisitos de implantación	Catálogo de requisitos

Tabla 5.11. Actividades, tareas y productos del proceso DSI.

Proceso CSI: Construcción del Sistema de Información		
Actividades	**Tareas**	**Productos**
CSI1 Preparación del entorno de generación y construcción	CSI1.2 Preparación del entorno de construcción	Entorno de construcción
CSI2 Generación del código de los componentes y procedimientos	CSI2.1 Generación del código de componentes	Producto de software
CSI3 Ejecución de las pruebas unitarias	CSI3.1 Preparación del entorno de pruebas unitarias	Entorno de pruebas unitarias
	CSI3.2 Realización y evaluación de las pruebas unitarias	Resultado de las pruebas unitarias
CSI4 Ejecución de las pruebas de integración	CSI4.1 Preparación del entorno de las pruebas de integración	Entorno de pruebas de integración
	CSI4.2 Realización de las pruebas de integración	Resultado de las pruebas de integración
	CSI4.3 Evaluación del resultado de las pruebas de integración	Evaluación del resultado de las pruebas de integración
CSI5 Ejecución de las pruebas del sistema	CSI5.1 Preparación del entorno de las pruebas del sistema	Entorno de pruebas del sistema
	CSI5.2 Realización de las pruebas del sistema	Resultado de las pruebas del sistema
	CSI5.3 Evaluación del resultado de las pruebas del sistema	Evaluación del resultado de las pruebas del sistema
CSI6 Elaboración de los manuales de usuario	CSI6.1 Elaboración de los manuales de usuario	Producto software, manuales de usuario

Tabla 5.12. Actividades, tareas y productos del proceso CSI.

Proceso IAS: Implantación y Aceptación del Sistema		
Actividades	**Tareas**	**Productos**
IAS3 Incorporación del sistema al entorno de operación	IAS3.1 Preparación de la instalación	Incidencias de preparación de instalación
	IAS3.2 Realización de la instalación	Producto software instalado
IAS5 Pruebas de implantación del sistema	IAS5.1 Preparación de las pruebas de implantación	Plan de pruebas
	IAS5.2 Realización de las pruebas de implantación	Resultado de las pruebas de implantación
	IAS5.3 Evaluación del resultado de las pruebas de implantación	Evaluación del resultado de las pruebas de implantación
IAS6 Pruebas de aceptación del sistema	IAS6.1 Preparación de las pruebas de aceptación	Plan de pruebas
	IAS6.2 Realización de las pruebas de aceptación	Resultado de las pruebas de aceptación
	IAS6.3 Evaluación del resultado de las pruebas de aceptación	Evaluación del resultado de las pruebas de aceptación

Tabla 5.13. Actividades, tareas y productos del proceso IAS.

Proceso MSI: Mantenimiento de Sistemas de Información
El alcance de la tesis se limita a la obtención de la primera versión del sistema, sin contemplar la posibilidad de situar al mismo en un entorno productivo que pudiera generar peticiones de cambios por parte de los usuarios. Por esta razón no se tienen en cuenta actividades relacionadas al mantenimiento del sistema.

Tabla 5.14. Actividades, tareas y productos del proceso MSI.

Interfaz: Gestión de proyectos		
Actividades	**Tareas**	**Productos**
GPI1 Estimación de esfuerzo	GPI1.1 Identificación de elementos a desarrollar	Elementos a desarrollar
	GPI1.2 Cálculo del esfuerzo	Esfuerzo estimado
GPI2 Planificación	GPI2.1 Selección de la estrategia de desarrollo	Planificación general del proyecto
	GPI2.2 Selección de la estructura de actividades, tareas y productos	Planificación general del proyecto
	GPI2.3 Establecimiento del calendario de hitos y entregas	Planificación general del proyecto

Tabla 5.15. Actividades, tareas y productos de la interfaz Gestión de proyectos.

Interfaz: Gestión de la configuración		
Actividades	**Tareas**	**Productos**
EVS-GC1 Definición de los requisitos de gestión de configuración	EVS-GC1.1 Definición de los requisitos de gestión de configuración	Requisitos de gestión de configuración
EVS-GC2 Establecimiento del plan de gestión de la configuración	EVS-GC2.1 Definición del plan de gestión de la configuración	Plan de gestión de la configuración para el sistema de información
	EVS-GC2.2 Especificación del entorno tecnológico para la gestión de configuración	Plan de gestión de la configuración para el sistema de información
GC1 Identificación y registro de productos	GC1.1 Identificación y registro de los productos de los procesos en el sistema de gestión de la configuración	Registro de los productos creados o modificados
GC2 Identificación y registro del producto global	GC2.1 Registro den el sistema de la configuración del producto global de proceso	Registro del producto global

Tabla 5.16. Actividades, tareas y productos de la interfaz Gestión de la configuración.

Interfaz: Aseguramiento de la calidad		
Actividades	**Tareas**	**Productos**
EVS-CAL1 Identificación de las propiedades de calidad para el sistema	EVS-CAL1.3 Identificación de las propiedades de calidad	Propiedades de calidad
EVS-CAL2 Establecimiento del plan de aseguramiento de calidad	EVS-CAL2.2 Alcance del plan de aseguramiento de calidad	Plan de aseguramiento de calidad
EVS-CAL3 Adecuación del plan de aseguramiento de calidad a la solución	EVS-CAL3.1 Ajuste del plan de aseguramiento de calidad	Plan de aseguramiento de calidad
ASI-CAL1 Especificación inicial del plan de aseguramiento de calidad	ASI-CAL1.1 Definición del plan de aseguramiento de calidad para el sistema de información	Plan de aseguramiento de calidad
ASI-CAL2 Especificación detallada del plan de aseguramiento de calidad	ASI-CAL2.1 Contenido del plan de aseguramiento de calidad para el sistema de información	Plan de aseguramiento de calidad Dossier de aseguramiento de calidad
ASI-CAL3 Revisión del análisis de consistencia	ASI-CAL3.1 Revisión del catálogo de requisitos	Revisión de requisitos
	ASI-CAL3.2 Revisión de la consistencia entre productos	Revisión de consistencia
ASI-CAL4 Revisión del plan de pruebas	ASI-CAL4.1 Revisión del plan de pruebas	Revisión del plan de pruebas
DSI-CAL1 Revisión de la verificación de la arquitectura del sistema	DSI-CAL1.1 Revisión de la consistencia entre productos del diseño	Revisión de la arquitectura del sistema
DSI-CAL2 Revisión de la especificación técnica del plan de pruebas	DSI-CAL2.1 Revisión del diseño de las pruebas unitarias, de integración y del sistema	Revisión del diseño de las pruebas
	DSI-CAL2.2 Revisión del plan de pruebas	Revisión del plan de pruebas

Interfaz: Aseguramiento de la calidad		
Actividades	**Tareas**	**Productos**
DSI-CAL3 Revisión de los requisitos de implantación	DSI-CAL3.1 Revisión de los requisitos de documentación de usuario	Revisión de los requisitos de documentación de usuario
	DSI-CAL3.2 Revisión de los requisitos de implantación	Revisión de los requisitos de implantación
CSI-CAL2 Revisión de las pruebas unitarias, de integración y del sistema	CSI-CAL2.1 Revisión de la realización de las pruebas unitarias	Revisión de la realización de las pruebas unitarias
	CSI-CAL2.2 Revisión de la realización de las pruebas de integración	Revisión de la realización de las pruebas de integración
	CSI-CAL2.3 Revisión de la realización de las pruebas del sistema	Revisión de la realización de las pruebas del sistema
CSI-CAL3 Revisión de los manuales de usuario	CSI-CAL3.1 Revisión de los manuales de usuario	Revisión de los manuales de usuario
IAS-CAL2 Revisión de las pruebas de implantación del sistema	IAS-CAL2.1 Revisión de la realización de las pruebas de implantación del sistema	Revisión de las pruebas de implantación del sistema
IAS-CAL3 Revisión de las pruebas de aceptación del sistema	IAS-CAL3.1 Revisión de la realización de las pruebas de aceptación del sistema	Revisión de las pruebas de aceptación del sistema

Tabla 5.17. Actividades, tareas y productos de la interfaz Aseguramiento de la calidad.

Interfaz: Seguridad		
Actividades	**Tareas**	**Productos**
EVS-SEG5 Evaluación detallada de la seguridad de la solución propuesta	EVS-SEG5.1 Descripción detallada de la seguridad de la solución propuesta	Seguridad de la solución propuesta

Tabla 5.18. Actividades, tareas y productos de la interfaz Seguridad.

5.1.2.3 Calendario de actividades, hitos y entregas

Teniendo en cuenta la estimación del esfuerzo llevada a cabo, la estrategia de desarrollo seleccionada, y las actividades, tareas y productos de la metodología indicados en el apartado anterior, se elaboró un plan en el cual se detallan las iteraciones, procesos y actividades, remarcando los hitos fundamentales del proyecto.

A continuación se muestra el plan desde lo más general a lo más específico, de manera de facilitar su comprensión.

En el nivel más general, el plan comprende cuatro iteraciones como se muestra en la figura 5.2.

ID	Nombre de tarea	Duration	Start	Finish
1	**Iteración 1 - Viabilidad**	**42 days**	**Sat 6/21/03**	**Fri 8/1/03**
19	**Iteración 2 - Arquitectura**	**145 days**	**Sat 8/2/03**	**Wed 12/24/03**
54	**Iteración 3 - Construcción**	**36 days**	**Thu 12/25/03**	**Thu 1/29/04**
86	**Iteración 4 - Cierre**	**19 days**	**Fri 1/30/04**	**Tue 2/17/04**

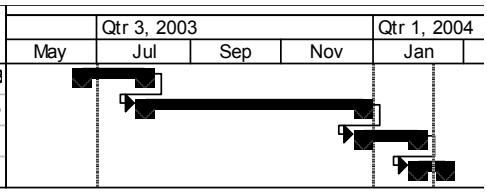

Figura 5.2. Iteraciones con sus fechas de comienzo y finalización.

Durante cada iteración se llevan a cabo los procesos e interfaces de la metodología, como se muestra en la figura 5.3. Además, el final de cada iteración se marca con un hito de entrega de productos, donde se entregan todos los productos generados o actualizados durante la iteración.

Todos los productos resultantes de las iteraciones se consolidarán al final del proyecto para dar forma al documento de tesis. Por tal motivo, y con el fin de mantener la claridad de la lectura, se indica entre paréntesis el capítulo del documento de tesis donde se alojará cada uno de los productos.

ID	Nombre de tarea	Duration	Start	Finish
1	**Iteración 1 - Viabilidad**	**42 days**	**Sat 6/21/03**	**Fri 8/1/03**
2	**Proceso EVS**	**19 days**	**Sat 6/21/03**	**Wed 7/9/03**
6	**Interfaz Gestión de proyectos**	**14 days**	**Thu 7/10/03**	**Wed 7/23/03**
9	**Interfaz Gestión de la configuración**	**4 days**	**Thu 7/24/03**	**Sun 7/27/03**
12	**Interfaz Aseguramiento de la calidad**	**3 days**	**Mon 7/28/03**	**Wed 7/30/03**
16	**Interfaz Seguridad**	**1 day**	**Thu 7/31/03**	**Thu 7/31/03**
18	Entrega de productos	1 day	Fri 8/1/03	Fri 8/1/03
19	**Iteración 2 - Arquitectura**	**145 days**	**Sat 8/2/03**	**Wed 12/24/03**
20	**Proceso ASI**	**32 days**	**Sat 8/2/03**	**Tue 9/2/03**
29	**Proceso DSI**	**59 days**	**Wed 9/3/03**	**Fri 10/31/03**
37	**Proceso CSI**	**49 days**	**Mon 11/3/03**	**Sun 12/21/03**
43	**Interfaz Gestión de la configuración**	**59 days**	**Fri 9/26/03**	**Sun 11/23/03**
45	**Interfaz Aseguramiento de la calidad**	**144 days**	**Sat 8/2/03**	**Tue 12/23/03**
53	Entrega de productos	1 day	Wed 12/24/03	Wed 12/24/03
54	**Iteración 3 - Construcción**	**36 days**	**Thu 12/25/03**	**Thu 1/29/04**
55	**Proceso ASI**	**7 days**	**Thu 12/25/03**	**Wed 12/31/03**
62	**Proceso DSI**	**8 days**	**Thu 1/1/04**	**Thu 1/8/04**
69	**Proceso CSI**	**14 days**	**Fri 1/9/04**	**Thu 1/22/04**
74	**Interfaz Gestión de la configuración**	**35 days**	**Thu 12/25/03**	**Wed 1/28/04**
77	**Interfaz Aseguramiento de la calidad**	**25 days**	**Tue 12/30/03**	**Fri 1/23/04**
85	Entrega de productos	1 day	Thu 1/29/04	Thu 1/29/04
86	**Iteración 4 - Cierre**	**19 days**	**Fri 1/30/04**	**Tue 2/17/04**
87	**Proceso CSI**	**5 days**	**Fri 1/30/04**	**Tue 2/3/04**
89	**Proceso IAS**	**8 days**	**Wed 2/4/04**	**Wed 2/11/04**
93	**Interfaz Gestión de la configuración**	**18 days**	**Fri 1/30/04**	**Mon 2/16/04**
96	**Interfaz Aseguramiento de la calidad**	**9 days**	**Wed 2/4/04**	**Thu 2/12/04**
100	Entrega de productos	1 day	Tue 2/17/04	Tue 2/17/04

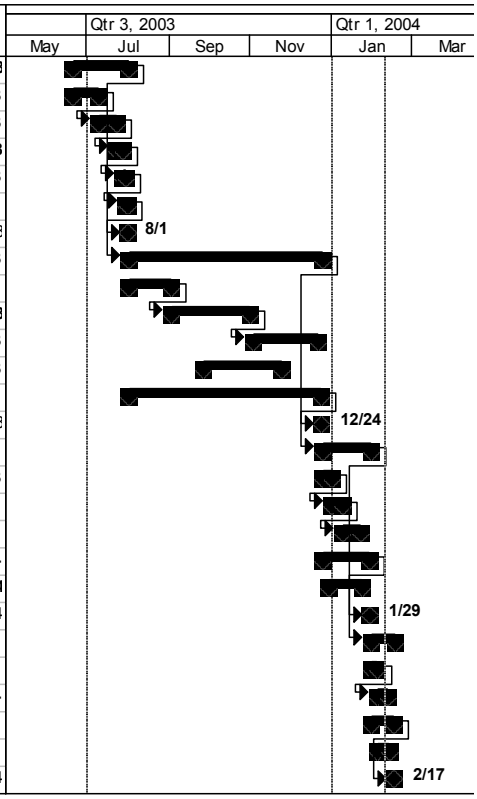

Figura 5.3. Procesos e hitos de cada iteración.

Los criterios seguidos para dimensionar las iteraciones fueron los siguientes:

- Por un lado, tomando un criterio descendente, se aplicaron porcentajes de esfuerzo a cada iteración, de acuerdo a las actividades involucradas. Los resultados fueron los siguientes:
  - Para la iteración de Viabilidad se toma un esfuerzo de aproximadamente el 15% del total.
  - Para la iteración de Arquitectura se toma un esfuerzo entre el 50% y el 60% del total.
  - Para la iteración de Construcción se toma un esfuerzo de aproximadamente el 15% del total.
  - Para la iteración de Cierre se toma un esfuerzo de aproximadamente el 10% del total.
- Por otro lado, tomando un criterio ascendente, se realizó una planificación detallada de cada iteración, arribando a porcentajes similares.

Los resultados de ambas aproximaciones se fusionaron, llegando a los planes que se muestran a continuación.

Iteración 1 – Viabilidad

La figura 5.4 muestra el detalle de procesos y actividades de esta iteración.

ID	Nombre de tarea	Duration	Start	Finish
1	**Iteración 1 - Viabilidad**	**42 days**	**Sat 6/21/03**	**Fri 8/1/03**
2	**Proceso EVS**	**19 days**	**Sat 6/21/03**	**Wed 7/9/03**
3	EVS1 Establecimiento del alcance del sistema	15 days	Sat 6/21/03	Sat 7/5/03
4	EVS2 Estudio de la situación actual	4 days	Sun 7/6/03	Wed 7/9/03
5	EVS3 Definición de requisitos del sistema	4 days	Sun 7/6/03	Wed 7/9/03
6	**Interfaz Gestión de proyectos**	**14 days**	**Thu 7/10/03**	**Wed 7/23/03**
7	GPI1 Estimación de esfuerzo	4 days	Thu 7/10/03	Sun 7/13/03
8	GPI2 Planificación	10 days	Mon 7/14/03	Wed 7/23/03
9	**Interfaz Gestión de la configuración**	**4 days**	**Thu 7/24/03**	**Sun 7/27/03**
10	EVS-GC1 Definición de los requisitos de gestión de configuración	1 day	Thu 7/24/03	Thu 7/24/03
11	EVS-GC2 Establecimiento del plan de gestión de la configuración	3 days	Fri 7/25/03	Sun 7/27/03
12	**Interfaz Aseguramiento de la calidad**	**3 days**	**Mon 7/28/03**	**Wed 7/30/03**
13	EVS-CAL1 Identificación de las propiedades de calidad para el sistema	1 day	Mon 7/28/03	Mon 7/28/03
14	EVS-CAL2 Establecimiento del plan de aseguramiento de calidad	1 day	Tue 7/29/03	Tue 7/29/03
15	EVS-CAL3 Adecuación del plan de aseguramiento de calidad a la solución	1 day	Wed 7/30/03	Wed 7/30/03
16	**Interfaz Seguridad**	**1 day**	**Thu 7/31/03**	**Thu 7/31/03**
17	EVS-SEG5 Evaluación detallada de la seguridad de la solución propuesta	1 day	Thu 7/31/03	Thu 7/31/03
18	Entrega de productos	1 day	Fri 8/1/03	Fri 8/1/03

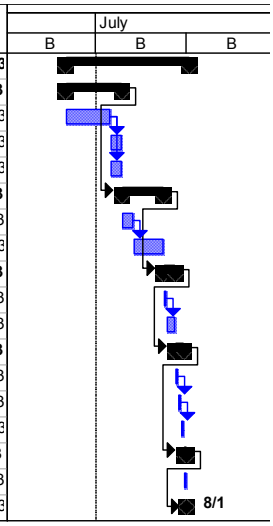

Figura 5.4. Detalle de procesos y actividades la Iteración 1.

Productos a entregar:

- Documento Estudio de viabilidad del sistema (integrado en Capítulo 3)
- Documento Plan general del proyecto (integrado en Capítulo 5)

- Gantt Bitácora del proyecto (integrado en Capítulo 13)
- Gantt Plan general del proyecto (integrado en Capítulo 5)
- Documento Catálogo de requisitos (integrado en Capítulo 4)
- Modelo en Enterprise Architect (integrado en Capítulos 3 al 7)

Iteración 2 – Arquitectura

La figura 5.5 muestra el detalle de procesos y actividades de esta iteración.

ID	Nombre de tarea	Duration	Start	Finish
19	**Iteración 2 - Arquitectura**	**145 days**	**Sat 8/2/03**	**Wed 12/24/03**
20	**Proceso ASI**	**32 days**	**Sat 8/2/03**	**Tue 9/2/03**
21	ASI1 Definición del sistema	7 days	Sat 8/2/03	Fri 8/8/03
22	ASI2 Establecimiento de requisitos	5 days	Sat 8/9/03	Wed 8/13/03
23	ASI8 Definición de interfaces de usuario	5 days	Thu 8/14/03	Mon 8/18/03
24	ASI3 Identificación de subsistemas de análisis	1 day	Sat 8/16/03	Sat 8/16/03
25	ASI4 Análisis de los casos de uso	7 days	Tue 8/19/03	Mon 8/25/03
26	ASI5 Análisis de las clases	7 days	Tue 8/19/03	Mon 8/25/03
27	ASI9 Análisis de consistencia y especificación de requisitos	5 days	Tue 8/26/03	Sat 8/30/03
28	ASI10 Especificación del plan de pruebas	3 days	Sun 8/31/03	Tue 9/2/03
29	**Proceso DSI**	**59 days**	**Wed 9/3/03**	**Fri 10/31/03**
30	DSI1 Definición de la arquitectura del sistema	5 days	Wed 9/3/03	Sun 9/7/03
31	DSI2 Diseño de la arquitectura de soporte	5 days	Mon 9/8/03	Fri 9/12/03
32	DSI3 Diseño de casos de uso reales	20 days	Sat 9/13/03	Fri 10/10/03
33	DSI4 Diseño de clases	20 days	Sat 9/13/03	Fri 10/10/03
34	DSI7 Verificación y aceptación de la arquitectura del sistema	2 days	Sat 10/11/03	Tue 10/14/03
35	DSI8 Generación de especificaciones de construcción	2 days	Wed 10/15/03	Thu 10/16/03
36	DSI10 Especificación técnica del plan de pruebas	11 days	Fri 10/17/03	Fri 10/31/03
37	**Proceso CSI**	**49 days**	**Mon 11/3/03**	**Sun 12/21/03**
38	CSI1 Preparación del entorno de generación y construcción	2 days	Mon 11/3/03	Tue 11/4/03
39	CSI2 Generación del código de los componentes y procedimientos	30 days	Wed 11/5/03	Tue 12/16/03
40	CSI3 Ejecución de las pruebas unitarias	30 days	Wed 11/5/03	Tue 12/16/03
41	CSI4 Ejecución de las pruebas de integración	30 days	Wed 11/5/03	Tue 12/16/03
42	CSI5 Ejecución de las pruebas del sistema	3 days	Wed 12/17/03	Sun 12/21/03
43	**Interfaz Gestión de la configuración**	**59 days**	**Fri 9/26/03**	**Sun 11/23/03**
44	GC1 Identificación y registro de productos	41 days	Fri 9/26/03	Sun 11/23/03
45	**Interfaz Aseguramiento de la calidad**	**144 days**	**Sat 8/2/03**	**Tue 12/23/03**
46	ASI-CAL1 Especificación inicial del plan de aseguramiento de calidad	4 days	Sat 8/2/03	Tue 8/5/03
47	ASI-CAL2 Especificación detallada del plan de aseguramiento de calidad	2 days	Tue 8/19/03	Wed 8/20/03
48	ASI-CAL3 Revisión del análisis de consistencia	1 day	Sun 8/31/03	Sun 8/31/03
49	ASI-CAL4 Revisión del plan de pruebas	1 day	Wed 9/3/03	Wed 9/3/03
50	DSI-CAL1 Revisión de la verificación de la arquitectura del sistema	1 day	Wed 10/15/03	Wed 10/15/03
51	DSI-CAL2 Revisión de la especificación técnica del plan de pruebas	0 days	Sat 11/1/03	Sat 11/1/03
52	CSI-CAL2 Revisión de las pruebas unitarias, de integración y del sistema	2 days	Mon 12/22/03	Tue 12/23/03
53	Entrega de productos	1 day	Wed 12/24/03	Wed 12/24/03

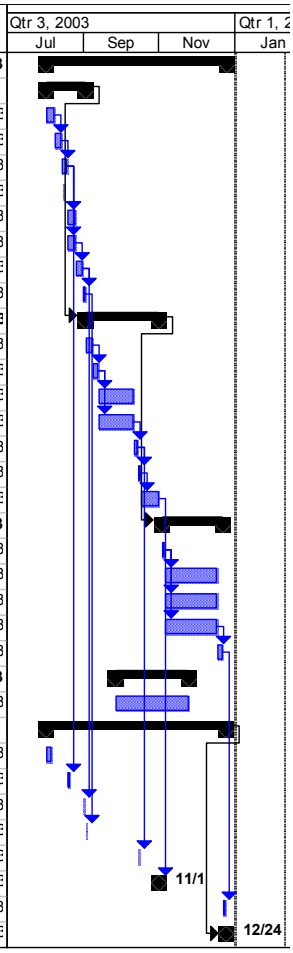

Figura 5.5. Detalle de procesos y actividades la Iteración 2.

Productos a entregar:

- Documento Análisis del sistema (integrado en Capítulo 6)
- Documento Diseño del sistema (integrado en Capítulo 7)
- Documento Construcción del sistema (integrado en Capítulo 9)
- Documento Catálogo de requisitos (integrado en Capítulo 3)
- Documento Modelo de casos de uso (integrado en Capítulo 3)
- Documento Modelo de análisis (integrado en Capítulo 6)

- Documento Modelo de diseño (integrado en Capítulo 7)
- Dossier de aseguramiento de calidad (integrado en Capítulo 13)
- Producto ejecutable (primera versión de la herramienta)
- Documento Plan general del proyecto (integrado en Capítulo 5)
- Gantt Bitácora del proyecto (integrado en Capítulo 13)
- Modelo en Enterprise Architect (integrado en Capítulos 3 al 7)

Iteración 3 – Construcción

La figura 5.6 muestra el detalle de procesos y actividades de esta iteración.

ID	Nombre de tarea	Duration	Start	Finish
54	**Iteración 3 - Construcción**	**36 days**	**Thu 12/25/03**	**Thu 1/29/04**
55	**Proceso ASI**	**7 days**	**Thu 12/25/03**	**Wed 12/31/03**
56	ASI2 Establecimiento de requisitos	2 days	Thu 12/25/03	Fri 12/26/03
57	ASI4 Análisis de los casos de uso	1 day	Sat 12/27/03	Mon 12/29/03
58	ASI5 Análisis de las clases	1 day	Sat 12/27/03	Mon 12/29/03
59	ASI8 Definición de interfaces de usuario	1 day	Sat 12/27/03	Mon 12/29/03
60	ASI9 Análisis de consistencia y especificación de requisitos	1 day	Tue 12/30/03	Tue 12/30/03
61	ASI10 Especificación del plan de pruebas	1 day	Wed 12/31/03	Wed 12/31/03
62	**Proceso DSI**	**8 days**	**Thu 1/1/04**	**Thu 1/8/04**
63	DSI3 Diseño de casos de uso reales	2 days	Thu 1/1/04	Sat 1/3/04
64	DSI4 Diseño de clases	2 days	Thu 1/1/04	Sat 1/3/04
65	DSI7 Verificación y aceptación de la arquitectura del sistema	1 day	Mon 1/5/04	Mon 1/5/04
66	DSI8 Generación de especificaciones de construcción	1 day	Tue 1/6/04	Tue 1/6/04
67	DSI10 Especificación técnica del plan de pruebas	1 day	Wed 1/7/04	Wed 1/7/04
68	DSI11 Establecimiento de requisitos de implantación	1 day	Thu 1/8/04	Thu 1/8/04
69	**Proceso CSI**	**14 days**	**Fri 1/9/04**	**Thu 1/22/04**
70	CSI2 Generación del código de los componentes y procedimientos	8 days	Fri 1/9/04	Tue 1/20/04
71	CSI3 Ejecución de las pruebas unitarias	8 days	Fri 1/9/04	Tue 1/20/04
72	CSI4 Ejecución de las pruebas de integración	8 days	Fri 1/9/04	Tue 1/20/04
73	CSI5 Ejecución de las pruebas del sistema	2 days	Wed 1/21/04	Thu 1/22/04
74	**Interfaz Gestión de la configuración**	**35 days**	**Thu 12/25/03**	**Wed 1/28/04**
75	GC1 Identificación y registro de productos	24 days	Thu 12/25/03	Tue 1/27/04
76	GC2 Identificación y registro del producto global	1 day	Wed 1/28/04	Wed 1/28/04
77	**Interfaz Aseguramiento de la calidad**	**25 days**	**Tue 12/30/03**	**Fri 1/23/04**
78	ASI-CAL2 Especificación detallada del plan de aseguramiento de calidad	1 day	Tue 12/30/03	Tue 12/30/03
79	ASI-CAL3 Revisión del análisis de consistencia	1 day	Wed 12/31/03	Wed 12/31/03
80	ASI-CAL4 Revisión del plan de pruebas	1 day	Thu 1/1/04	Thu 1/1/04
81	DSI-CAL1 Revisión de la verificación de la arquitectura del sistema	1 day	Tue 1/6/04	Tue 1/6/04
82	DSI-CAL2 Revisión de la especificación técnica del plan de pruebas	1 day	Thu 1/8/04	Thu 1/8/04
83	DSI-CAL3 Revisión de los requisitos de implantación	1 day	Fri 1/9/04	Fri 1/9/04
84	CSI-CAL2 Revisión de las pruebas unitarias, de integración y del sistema	1 day	Fri 1/23/04	Fri 1/23/04
85	Entrega de productos	1 day	Thu 1/29/04	Thu 1/29/04

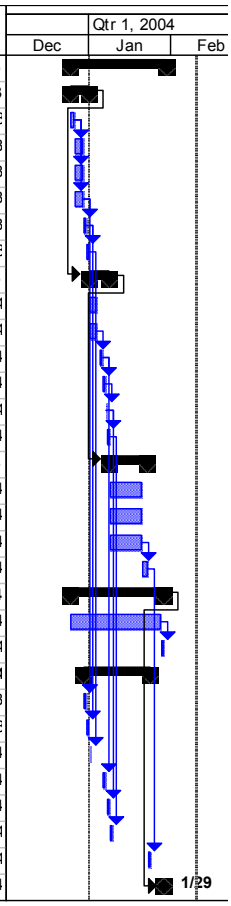

Figura 5.6. Detalle de procesos y actividades la Iteración 3.

Productos a entregar:

- Documento Análisis del sistema (integrado en Capítulo 6)
- Documento Diseño del sistema (integrado en Capítulo 7)
- Documento Construcción del sistema (integrado en Capítulo 9)
- Documento Catálogo de requisitos (integrado en Capítulo 3)
- Documento Modelo de casos de uso (integrado en Capítulo 3)

- Documento Modelo de análisis (integrado en Capítulo 6)
- Documento Modelo de diseño (integrado en Capítulo 7)
- Dossier de aseguramiento de calidad (integrado en Capítulo 13)
- Producto ejecutable (versión beta de la herramienta)
- Documento Plan general del proyecto (integrado en Capítulo 5)
- Gantt Bitácora del proyecto (integrado en Capítulo 13)
- Modelo en Enterprise Architect (integrado en Capítulos 3 al 7)

Iteración 4 – Cierre

La figura 5.7 muestra el detalle de procesos y actividades de esta iteración.

ID	Nombre de tarea	Duration	Start	Finish
86	**Iteración 4 - Cierre**	**19 days**	**Fri 1/30/04**	**Tue 2/17/04**
87	**Proceso CSI**	**5 days**	**Fri 1/30/04**	**Tue 2/3/04**
88	CSI6 Elaboración de los manuales de usuario	3 days	Fri 1/30/04	Tue 2/3/04
89	**Proceso IAS**	**8 days**	**Wed 2/4/04**	**Wed 2/11/04**
90	IAS3 Incorporación del sistema al entorno de operación	1 day	Wed 2/4/04	Wed 2/4/04
91	IAS5 Pruebas de implantación del sistema	3 days	Wed 2/4/04	Sat 2/7/04
92	IAS6 Pruebas de aceptación del sistema	3 days	Sun 2/8/04	Wed 2/11/04
93	**Interfaz Gestión de la configuración**	**18 days**	**Fri 1/30/04**	**Mon 2/16/04**
94	GC1 Identificación y registro de productos	11 days	Fri 1/30/04	Sun 2/15/04
95	GC2 Identificación y registro del producto global	1 day	Mon 2/16/04	Mon 2/16/04
96	**Interfaz Aseguramiento de la calidad**	**9 days**	**Wed 2/4/04**	**Thu 2/12/04**
97	CSI-CAL3 Revisión de los manuales de usuario	2 days	Wed 2/4/04	Thu 2/5/04
98	IAS-CAL2 Revisión de las pruebas de implantación del sistema	0 days	Sun 2/8/04	Sun 2/8/04
99	IAS-CAL3 Revisión de las pruebas de aceptación del sistema	1 day	Thu 2/12/04	Thu 2/12/04
100	Entrega de productos	1 day	Tue 2/17/04	Tue 2/17/04

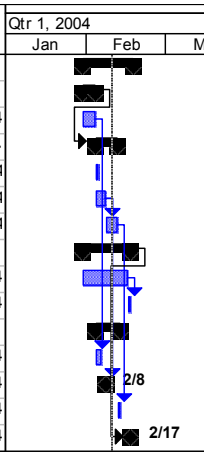

Figura 5.7. Detalle de procesos y actividades la Iteración 4.

Productos a entregar:

- Documento Manual del usuario (integrado en Capítulo 13)
- Documento Implantación del sistema (integrado en Capítulo 10)
- Dossier de aseguramiento de calidad (integrado en Capítulo 13)
- Producto ejecutable (release estable de la herramienta)
- Gantt Bitácora del proyecto (integrado en Capítulo 13)
- Modelo en Enterprise Architect (integrado en Capítulos 3 a 7)

5.2 Gestión de la configuración

5.2.1 Requisitos de gestión de la configuración

Se plantea la necesidad de gestionar la configuración de todos los elementos que constituyen el sistema objeto del presente trabajo de tesis.

En términos generales, se plantean los siguientes requisitos para la gestión de la configuración:

- Todos los elementos constitutivos del proyecto (documentos, modelos, código fuente, librerías, etc) deberán estar almacenados en un mismo repositorio.
- Deberán hacerse copias de resguardo del repositorio periódicamente.
- Cada uno de los elementos mencionados deberá encontrarse bajo control de versiones.
- Para cada cambio en cualquiera de los elementos, deberá guardarse la fecha y hora del cambio y una breve descripción del mismo.
- Se deberán trazar líneas base para marcar un conjunto de elementos en un determinado estado del proyecto (generalmente luego de finalizado un proceso durante el desarrollo del sistema).
- Se deberá poder retroceder a cualquier línea base recuperando todos los elementos del proyecto en el estado en el que estaban al momento de trazada la línea base.
- Se deberá poder efectuar una ramificación en los elementos del proyecto, con el fin de hacer desarrollo en paralelo de distintas alternativas (sobre todo durante el proceso de Construcción del sistema).
- Se deberán poder obtener listados o reportes con información sobre los cambios en los elementos (elementos que sufrieron más cambios, cambios entre dos líneas base, etc).

5.2.2 Plan de gestión de la configuración

5.2.2.1 Entorno tecnológico

Se adopta el producto *ClearCase LT* [ClearCase LT, 2004] como herramienta de gestión de la configuración. Las razones para la adopción de esta herramienta se basan en los siguientes fundamentos:

- Cubre ampliamente los requisitos de gestión de la configuración definidos en el apartado anterior.

- Es de rápida instalación y configuración.
- Es muy amigable para el usuario. La existencia del repositorio es prácticamente transparente para el mismo, quien puede interactuar directamente con archivos en disco como lo hace usualmente.
- Posee herramientas gráficas para visualización del árbol de versiones de cada elemento, para la creación de ramificaciones y para el trazado de líneas base.
- Su esquema de backup está basado en la copia de archivos y directorios del sistema operativo, lo que permite hacer resguardos en cualquier dispositivo existente.
- Posee un esquema de vistas configurables que permite visualizar el estado de todos los elementos remitiéndose al momento histórico (línea base) en que se desee.
- Cuenta con una herramienta para la generación de reportes vinculados a la gestión de configuraciones (elementos que sufrieron más cambios, cambios entre líneas base, etc).

ClearCase LT se instalará en la computadora personal a utilizar como plataforma para el desarrollo de la tesis, la cual cuenta con el siguiente hardware:

- Procesador Pentium III 700 MHz
- 384 MB de RAM
- Disco rígido de 12 GB

5.2.2.2 Identificación y ubicación de los elementos

Los elementos constitutivos del proyecto se agruparán en diferentes carpetas de acuerdo a las actividades realizadas a lo largo del proyecto. Cada elemento constituye un ítem de configuración identificado de manera única, que será versionado de manera de mantener su evolución a lo largo del desarrollo. El apartado 5.2.2.4 detalla las reglas utilizadas para el versionado de los ítems de configuración.

La tabla 5.19 muestra las carpetas y los elementos contenidos en cada una de ellas.

Carpeta	*Subcarpeta*	*Elemento*	*Descripción*	*Tipo*
Gestión del proyecto		Estudio de viabilidad del sistema.doc	Documento que contiene los productos generados durante las actividades del proceso de Estudio de Viabilidad del Sistema (EVS).	Documento de Microsoft Word
		Plan general del proyecto.doc	Documento que contiene los productos generados por las interfaces Gestión de proyectos, Gestión de la configuración, Aseguramiento de la calidad y Seguridad, aplicadas durante el proceso de Estudio de viabilidad del sistema.	Documento de Microsoft Word
		Plan general del proyecto.mpp	Plan de fases en las que se divide el proyecto.	Documento de Microsoft Project
		Bitácora.mpp	Bitácora en la que se documenta el avance del proyecto día por día.	Documento de Microsoft Project
Requerimientos	Prototipos de interfaz	Prototipos de interfaz	Prototipos de interfaces gráficas de usuario.	Archivos .java generados con NetBeans
		Catálogo de requisitos.doc	Documento que contiene los requisitos especificados para el sistema.	Documento Microsoft Word
Modelos	Reportes	Reportes de los modelos	Documentos con reportes generados por herramienta CASE.	Documentos Microsoft Word
		Cat.eap	Modelos de Requerimientos, Casos de uso, Análisis, Diseño, Implementación y Despliegue. Documentados en UML dentro de herramienta CASE.	Archivo de modelo generado con Enterprise Architect
Fuentes	builds	Aplicación compilada	Aplicación resultante de compilar todos los fuentes	Archivo .jar
	conf	Archivos de configuración	Archivos de configuración de parámetros y mensajes de la aplicación	Archivos XML
	lib	Librerías necesarias	Librerías necesarias para compilar el sistema	Archivos .jar
	src	Archivos de código fuente	Archivos de código fuente de todo el sistema, ordenado por paquetes	Archivos .java
	test	Archivos de código fuente para pruebas	Archivos de código fuente que automatizan las pruebas unitarias y de integración mediante JUnit	Archivos .java
Calidad	Revisiones	Dossier de aseguramiento de calidad.doc	Documento de revisión de los distintos elementos que componen el proyecto	Documento Microsoft Word
	Pruebas	Plan de pruebas.doc	Documentos con el plan de pruebas, incluyendo el diseño de los casos de prueba de nivel de sistema y de aceptación (Casos de prueba funcionales y de rendimiento)	Documento Microsoft Word
Documentación		Análisis del sistema.doc	Documento con el modelo de análisis del sistema completo	Documento Microsoft Word
		Diseño del sistema.doc	Documento con el modelo de diseño del sistema completo	Documento Microsoft Word
		Construcción del sistema.doc	Documento con la información de construcción del sistema, incluyendo el resultado de las pruebas unitarias, de integración y de sistema	Documento Microsoft Word
		Implantación y aceptación del sistema.doc	Documento con la información de puesta en producción del sistema, incluyendo el resultado de las pruebas de instalación y aceptación	Documento Microsoft Word
		Manual del usuario.doc	Manual que comprende la instalación ,configuración, administración y uso del sistema.	Documento Microsoft Word
	Material de referencia	Documentos varios	Documentos con especificaciones de CMMI y SCAMPI	Documentos PDF
Documento de Tesis		Capítulos del documento	Documentos con cada uno de los capítulos que conforman el documento de tesis (extractados de la documentación anterior)	Documentos Microsoft Word

Tabla 5.19. Elementos constitutivos del proyecto, almacenados en el repositorio de gestión de la configuración.

La figura 5.8 muestra la estructura real del repositorio dentro de la herramienta.

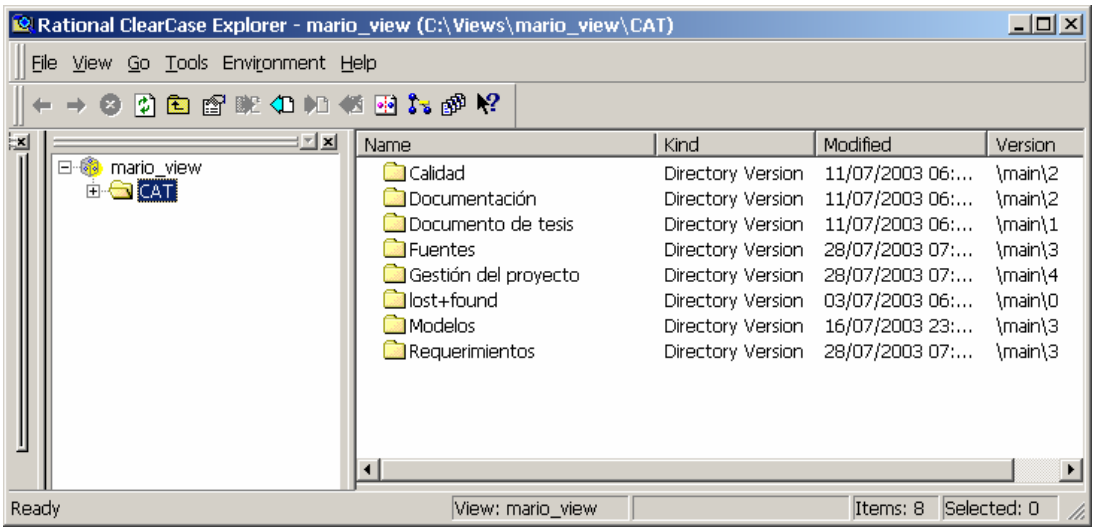

Figura 5.8: identificación y ubicación de los elementos en la estructura del repositorio. Cada elemento está identificado por su nombre y su versión.

5.2.2.3 Procedimientos

El gestor de configuraciones constituirá el repositorio donde se almacenen todos los elementos vinculados con el desarrollo del sistema, desde el inicio del proyecto hasta la finalización del mismo.

El procedimiento básico para la gestión de cada uno de los elementos es el siguiente:

- Cuando se crea un nuevo elemento, el mismo se agrega al repositorio de gestión de la configuración (operación "*add to source control*", donde permanece almacenado en modo sólo lectura. En el momento de agregarlo al repositorio, el gestor de configuraciones le asigna una identificación única que se mantendrá durante toda la vida del elemento. La identificación permanece oculta para el usuario, el cual visualiza el elemento simplemente por su nombre (nombre del archivo).
- Cuando es necesario modificar dicho elemento, se debe extraer el mismo del repositorio (operación "*checkout*"), quedando éste en modo lectura/escritura para que se le efectúen los cambios necesarios.
- Luego de efectuados los cambios, el elemento se inserta nuevamente en el repositorio (operación "*checkin*"). En este momento debe indicarse con un comentario cuáles fueron los cambios efectuados al elemento en cuestión.

- Luego del "*checkin*", queda almacenada en el repositorio una nueva versión del elemento modificado, con un comentario que indica cuáles fueron los cambios efectuados sobre el mismo.

Las figuras 5.9 a 5.11 muestran algunos ejemplos de estas operaciones dentro de la herramienta.

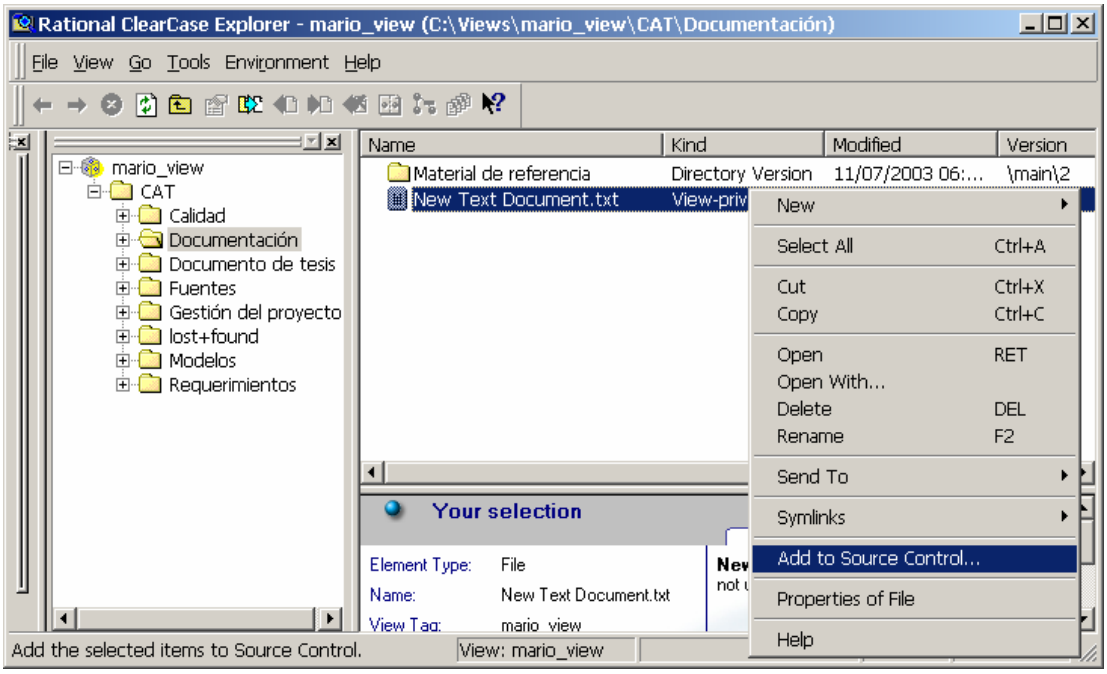

Figura 5.9. Operación "add to source control" (agregar al repositorio) sobre un elemento nuevo.

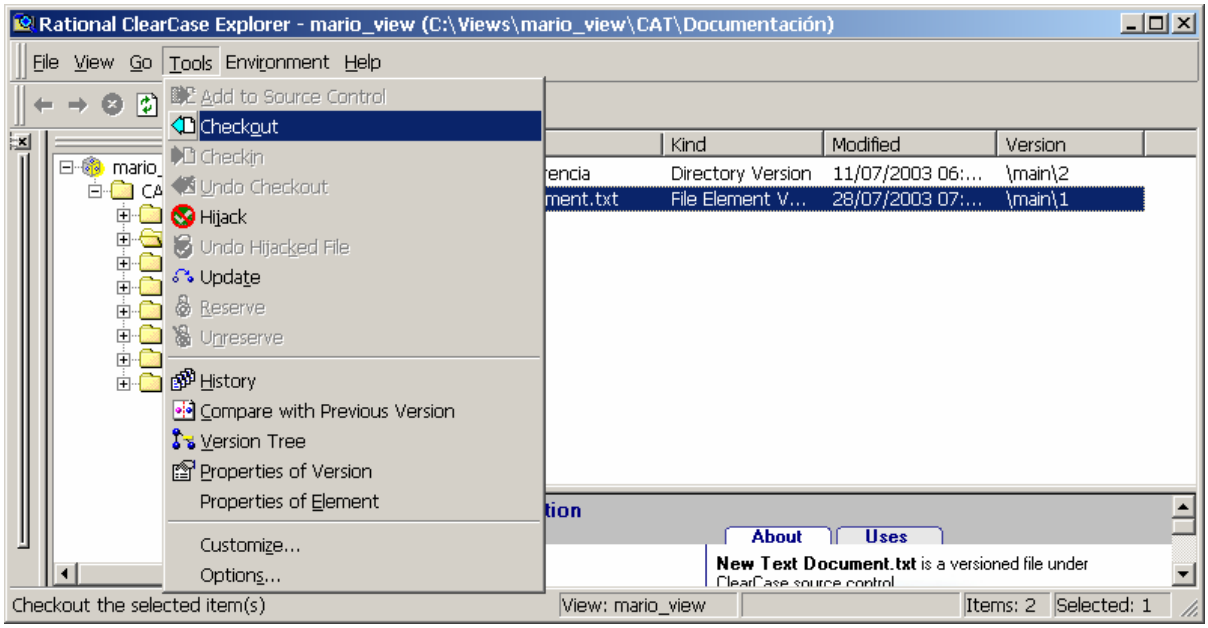

Figura 5.10a. Operación "checkout" (extracción) sobre un elemento versionado (parte 1).

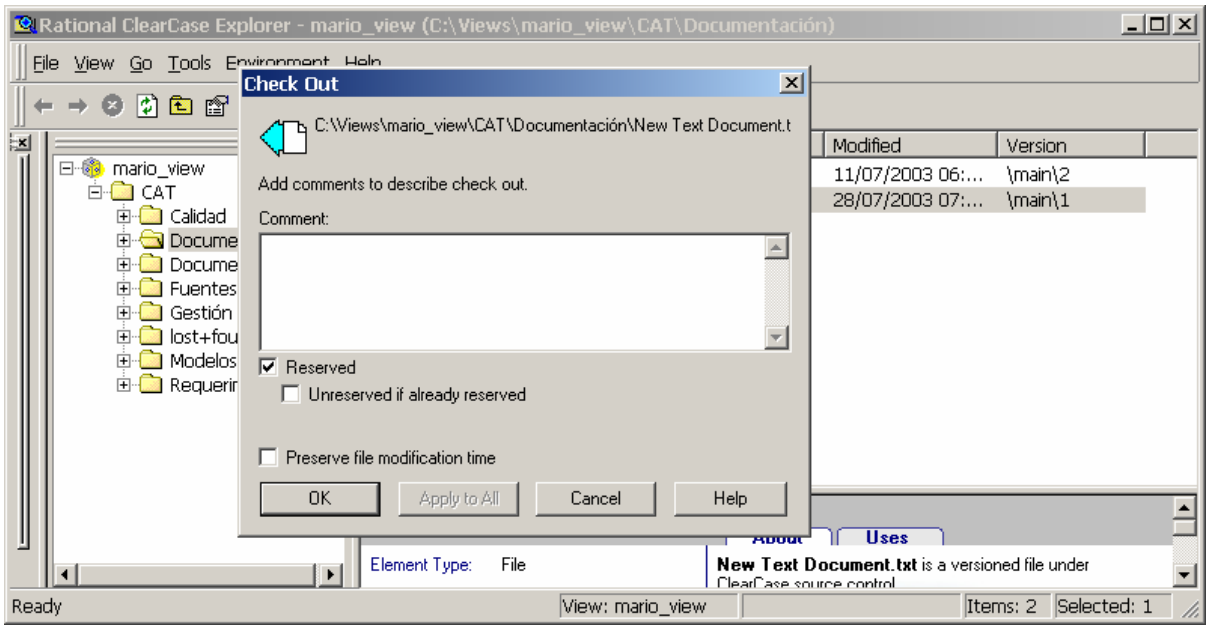

Figura 5.10b. Operación "checkout" (extracción) sobre un elemento versionado (parte 2).

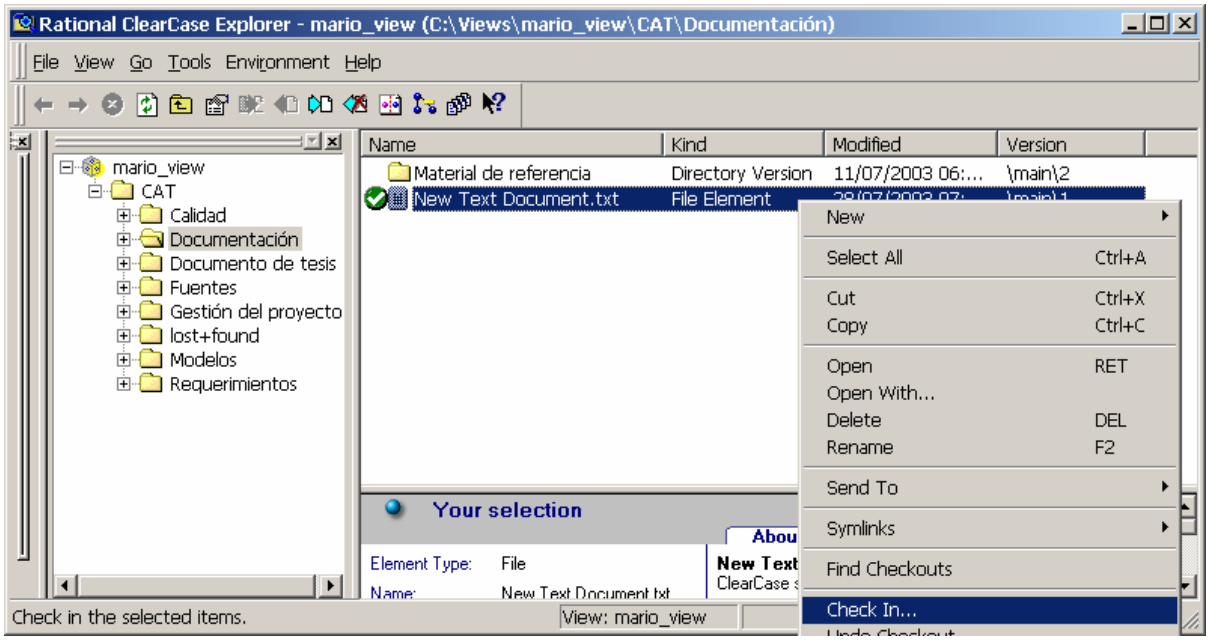

Figura 5.11. Operación "checkin" (inserción) sobre un elemento versionado.

5.2.2.4 Reglas de versionado

Los elementos del repositorio serán versionados de la siguiente manera:

- Al agregar el elemento en el repositorio se crea la rama principal del mismo y se genera la versión 1 del elemento contenida en esa rama.

- Con cada cambio ("*checkout*" y "*checkin*") que se efectúe sobre el elemento, se genera una nueva versión, incrementanto por unidades enteras (versiones 2, 3, 4, etc).
- Cuando se decide crear una versión ramificada de un elemento (operación "*branch*") se crea una nueva rama (secundaria) del mismo y se genera la versión 1 del elemento contenida en esa rama.
- Con cada cambio que se efectúe sobre la ramificación, se genera una nueva versión, incrementando por unidades enteras (al igual que en la rama principal).
- En cualquier momento se puede decidir fusionar la versión de la rama principal con la de la rama secundaria (operación "*merge*").

La figura 5.12 muestra una de las alternativas de la herramienta para visualizar los cambios efectuados en un elemento.

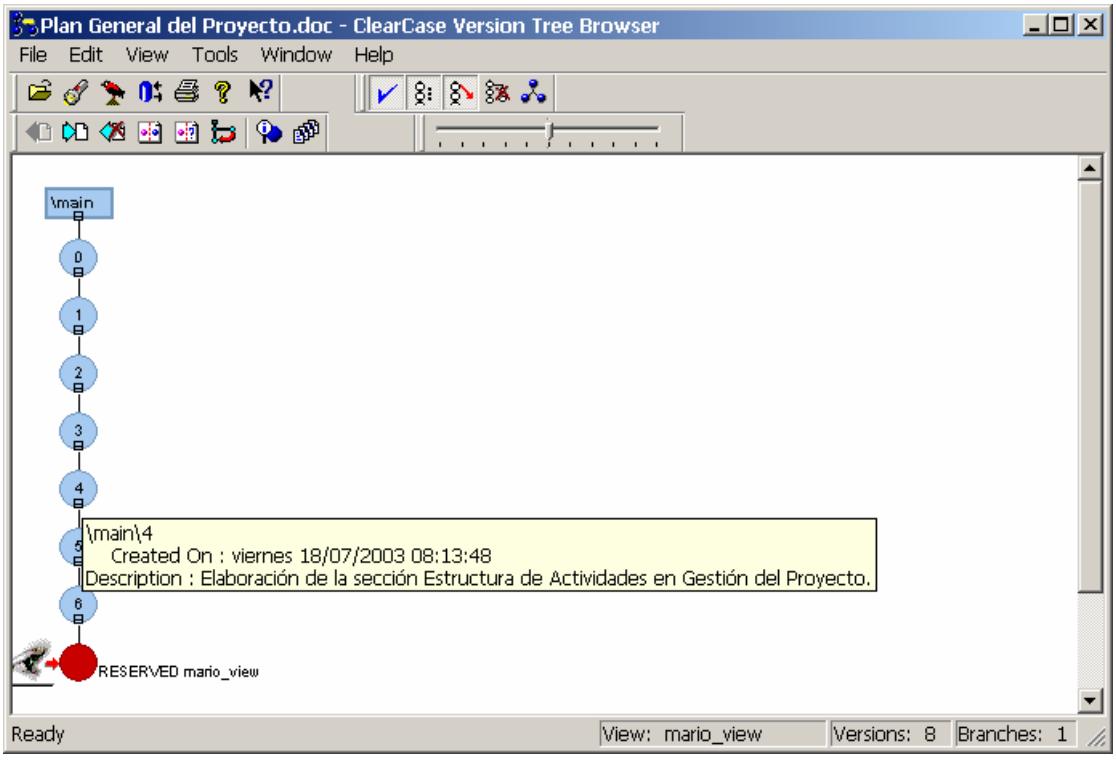

Figura 5.12. Herramienta "*version tree browser*" (visualizador de versiones) que permite visualizar los cambios efectuados en un elemento. El árbol muestra todas versiones del elemento (archivo Estudio de la Viabilidad del Sistema.doc) con la información asociada a cada uno de sus cambios y cuál es la versión que se está utilizando actualmente. Las ramificaciones ("*branches*") se visualizan como ramas que se desprenden de alguno de los círculos de versión. Las uniones ("*merges*") se visualizan como uniones entre dos ramas diferentes.

5.2.2.5 Información para auditorías

Cuando se efectúen operaciones "*checkin*" sobre cualquier elemento del repositorio, se anexará un comentario a la misma. En el comentario se explicará qué es lo que se ha cambiado en el elemento, y la herramienta auditará automáticamente el usuario y la fecha y hora del cambio.

La herramienta cuenta con la capacidad de auditar el histórico de cambios de cada elemento y de generar reportes que analizan los cambios en uno o más elementos.

La figura 5.13 muestra otra de las facilidades con que cuenta la herramienta para auditar el histórico de cambios.

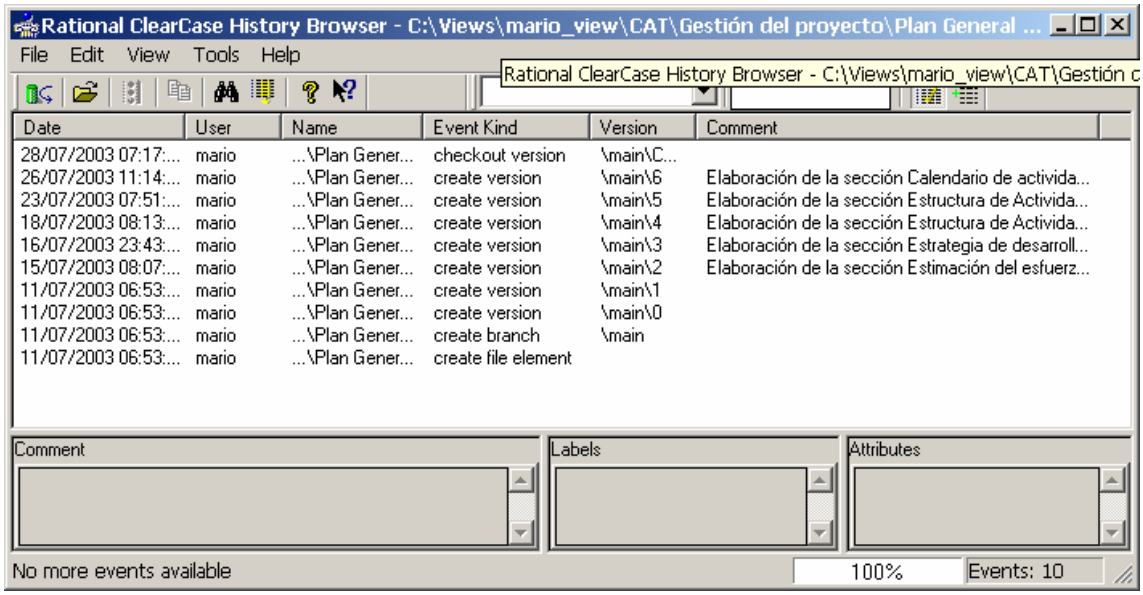

Figura 5.13. Herramienta "*history browser*" (visualizador de histórico de cambios) que permite auditar los cambios efectuados en un elemento.

5.3 Aseguramiento de la calidad

5.3.1 Propiedades de calidad

Para evaluar la calidad del sistema se tendrán en cuenta las siguientes propiedades:

- Correspondencia del sistema con los requisitos planteados
- Documentación de los distintos modelos del sistema (análisis, diseño, implementación y testing)
- Eficiencia
- Fiabilidad

- Facilidad de uso
- Facilidad de mantenimiento

5.3.2 Plan de aseguramiento de la calidad

5.3.2.1 Propósito y alcance

El propósito del plan de aseguramiento de la calidad es definir las actividades que se llevarán a cabo durante el proyecto con el fin de garantizar el cumplimiento de las propiedades de calidad definidas para el mismo.

Dichas actividades se realizarán a lo largo de todo el proyecto, retroalimentando constantemente el desarrollo de manera que las desviaciones sobre la calidad planificada sean corregidas apenas se detectan.

Las actividades contempladas por el plan consisten fundamentalmente de revisiones a efectuar durante el desarrollo del proyecto. Los resultados de dichas revisiones se documentan en el apartado 13.1 (Dossier de aseguramiento de la calidad) del Capítulo 13.

5.3.2.2 Plan de actividades

El plan de las actividades que se llevarán a cabo para el Aseguramiento de la calidad, se encuentra detallado en las secciones 5.1.2.2 (Estructura de actividades) y 5.1.2.3 (Calendario de actividades, hitos y entregas). Por tal motivo, no se repite la planificacion en este apartado.

5.4 Seguridad

5.4.1 Características de seguridad

Es sistema se plantea como una aplicación con las siguientes características:

- Es una aplicación del tipo *standalone* ejecutable en computadoras personales.
- No necesita ni comparte recursos de red.
- Es monousuario, es decir, no puede ser utilizada por más de un usuario a la vez.
- No posee niveles de permisos para acceder a la funcionalidad que expone al usuario.
- Almacena y recupera archivos en el disco local.

Dadas las características que presenta el sistema, se puede concluir que no es necesario contemplar mecanismos de seguridad tales como autenticación de usuarios,

autorización, niveles de acceso, seguridad ante transacciones concurrentes, o recuperación ante fallos en la infraestructura de comunicaciones (fallos de red).

Por otra parte, la capacidad de almacenar las evaluaciones en archivos desliga a la aplicación de la seguridad relacionada la pérdida de datos. Dicha seguridad queda a cargo del sistema operativo, o en última instancia del propio usuario quien deberá ocuparse de hacer backups periódicos de los archivos almacenados.

6. ANÁLISIS DEL SISTEMA

En este capítulo se documentan los productos de salida del proceso ASI (Análisis del Sistema de Información) de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

6.1 Modelos y trazabilidad

El proceso ASI (Análisis del Sistema de Información) cubre un conjunto de modelos relacionados entre sí. Este documento muestra cada uno de esos modelos mediante reportes extraídos de la herramienta *Enterprise Architect* [EA, 2004].

El diagrama de la figura 6.1 muestra las relaciones de trazabilidad que existen entre los distintos modelos presentados en el documento.

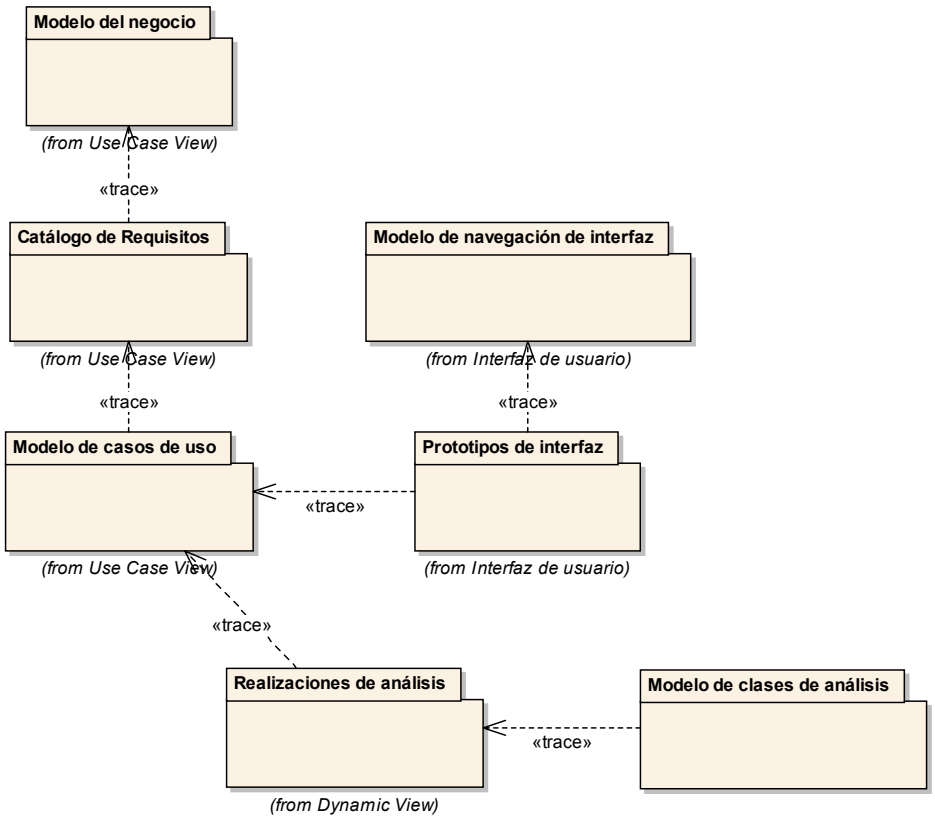

Figura 6.1. Trazabilidad entre los distintos modelos presentados en el documento.

6.2 Modelo del negocio

A continuación se muestra un reporte generado con la herramienta *Enterprise Architect*, mostrando el Modelo del negocio documentado en UML [Booch & Jacobson & Rumbaugh, 1998].

Reporte: Modelo del negocio

El modelo del negocio contempla los procesos principales del negocio bajo análisis y la forma en que los mismos se llevan a cabo. Dentro de este modelo, los procesos se representan mediante casos de uso del negocio. El detalle sobre las actividades llevadas a cabo y las entidades utilizadas para completar cada proceso, se documenta mediante diagramas de actividades.

El negocio cubierto por el sistema consiste básicamente en la evaluación del nivel de madurez de una organización de software de acuerdo al modelo CMMI-SW [CMMI-SW, 2002]. La figura 6.2 muestra conceptualmente la representación de este proceso en términos de casos de uso del negocio.

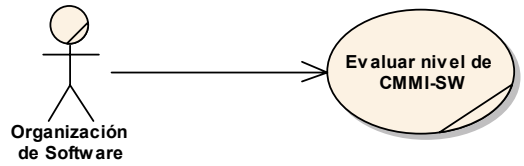

Figura 6.2. Actores y Casos de uso del negocio

Organización de Software

public «business actor» ***Actor:*** Este actor representa a la organización de software interesada en la evaluación de su nivel respecto al modelo CMMI-SW.

Evaluar nivel de CMMI-SW

public «business use case» ***Use Case:*** El proceso principal del negocio es la evaluación del nivel de CMMI-SW de la organización (o alguno de sus proyectos) de acuerdo al método SCAMPI [SCAMPI, 2001]. Como resultado del proceso se obtiene el nivel de la organización (o el proyecto) dentro del modelo CMMI-SW.

Las actividades llevadas a cabo para completar el proceso de evaluación descripto en este caso de uso se muestran en el diagrama de la figura 6.3.

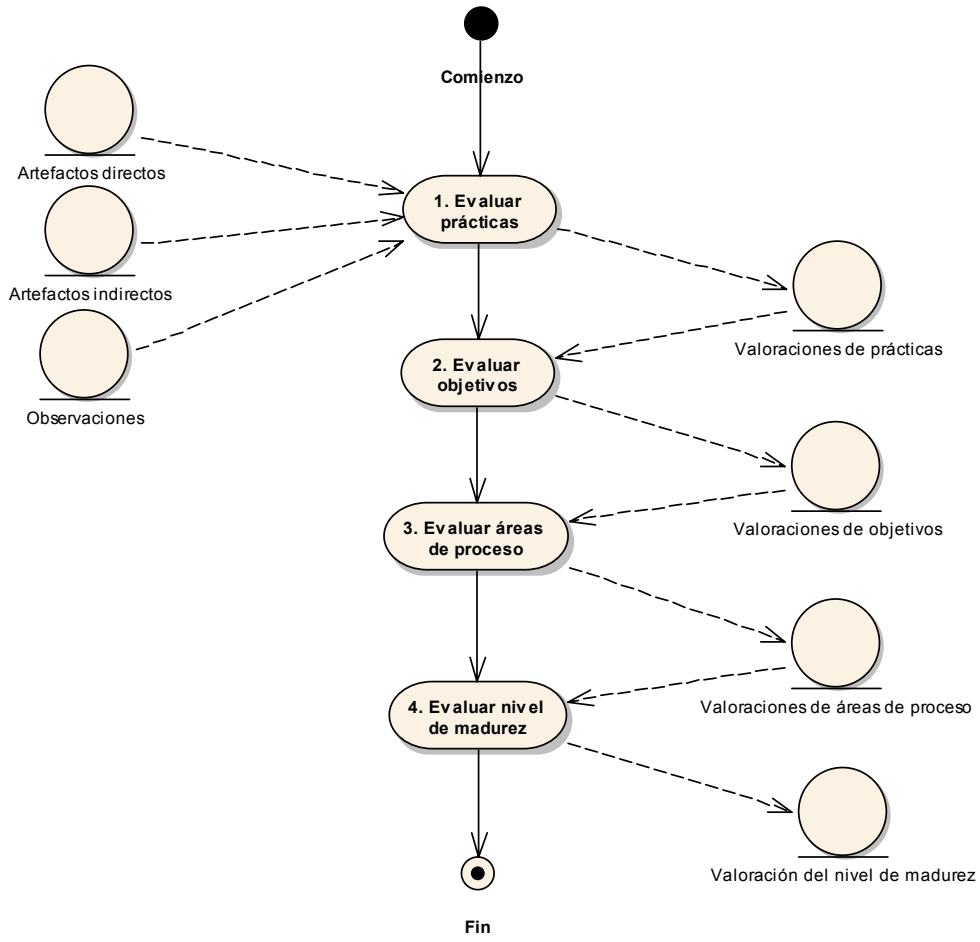

Figura 6.3. Este diagrama muestra las actividades llevadas a cabo en el caso de uso Evaluar nivel de CMMI-SW. Además de las actividades, se incluyen en el diagrama las entidades principales de entrada y de salida utilizadas durante el proceso.

A continuación se describen las actividades y las entidades contempladas en el diagrama.

1. Evaluar prácticas

public Activity: Esta actividad consiste en la evaluación de las prácticas del modelo CMMI-SW llevadas a cabo en la organización. Para la evaluación de las prácticas se tienen en cuenta los Artefactos directos, los Artefactos indirectos, y las Observaciones sobre las debilidades encontradas. Las prácticas se pueden evaluar a nivel de instancia (proyecto específico) o a nivel de conjunto de instancias (conjunto de proyectos). Como resultado de la actividad se obtienen las Valoraciones de prácticas.

El diagrama de la figura 6.4 muestra las reglas del negocio utilizadas para la evaluación de las prácticas.

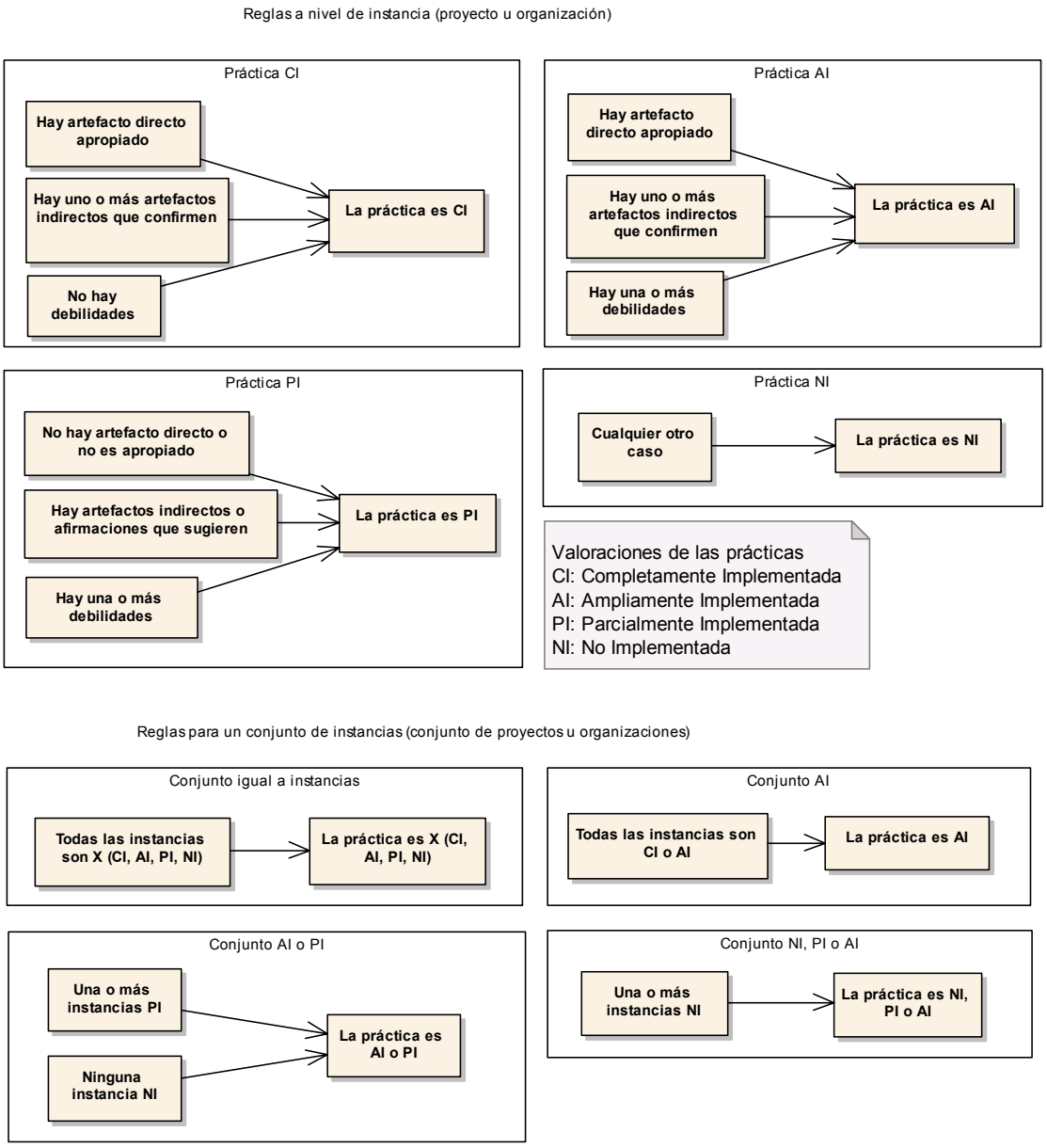

Figura 6.4. Reglas del negocio para la evaluación de las prácticas. Las reglas se muestran mediante una notación de Grafo Causal. Para cada regla, los rectángulos de la izquierda representan las condiciones y los de la derecha las conclusiones o acciones.

2. Evaluar objetivos

public Activity: Esta actividad consiste en la evaluación de los objetivos (genéricos y específicos) del modelo CMMI-SW alcanzados en la organización. Para la evaluación de los objetivos se tienen en cuenta los resultados de la evaluación de las prácticas asociadas al objetivo.

Como resultado de la actividad se obtienen las Valoraciones de objetivos.

El diagrama de la figura 6.5 muestra las reglas del negocio utilizadas para la evaluación de los objetivos.

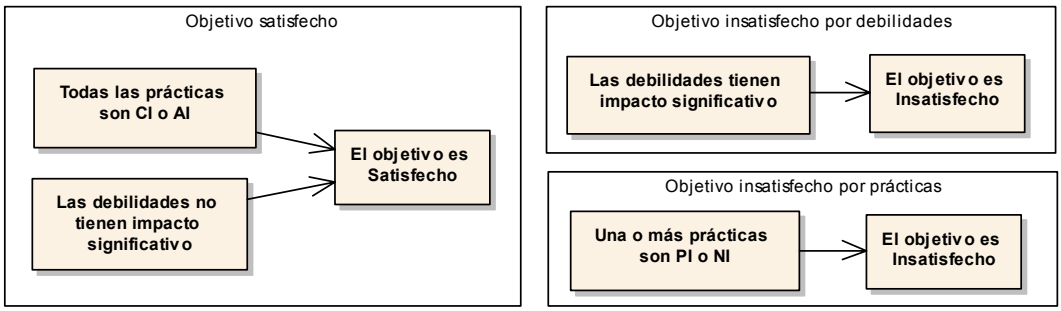

Figura 6.5. Reglas del negocio para la evaluación de los objetivos.

3. Evaluar áreas de proceso

public Activity: Esta actividad consiste en la evaluación de los niveles de satisfacción de las áreas de proceso del modelo CMMI-SW alcanzados en la organización. Para la evaluación de las áreas de proceso se tienen en cuenta los resultados de la evaluación de los objetivos asociados al área de proceso.

Como resultado de la actividad se obtienen las Valoraciones de áreas de proceso.

El diagrama de la figura 6.6 muestra las reglas del negocio utilizadas para la evaluación de las áreas de proceso.

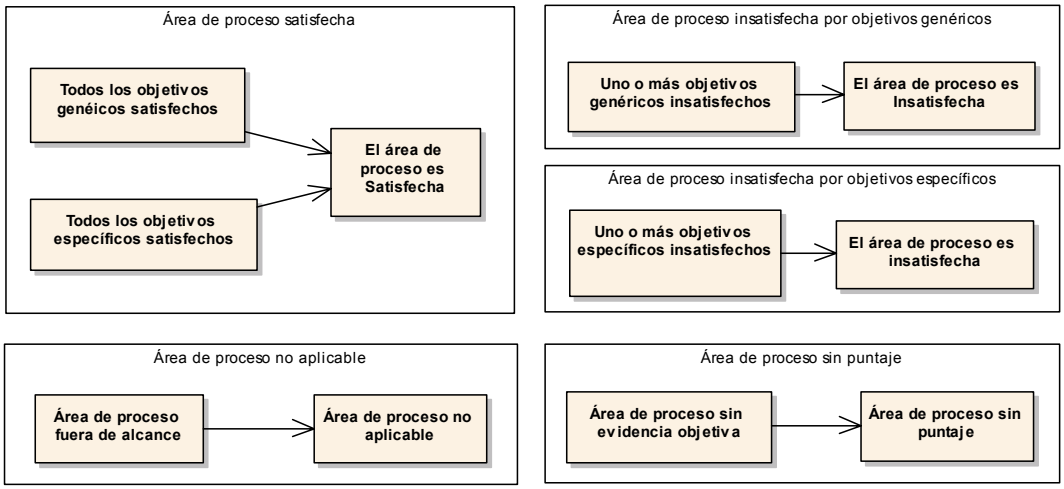

Figura 6.6. Reglas del negocio para la evaluación de las áreas de proceso

4. Evaluar nivel de madurez

public Activity: Esta actividad consiste en la evaluación del niveles de madurez del modelo CMMI-SW alcanzado en la organización. Para la evaluación del nivel de madurez se tienen en cuenta los resultados de la evaluación de las áreas de proceso asociadas al nivel de madurez.

Como resultado de la actividad se obtiene la Valoración del nivel de madurez.

El diagrama de la figura 6.7 muestra las reglas del negocio utilizadas para la evaluación de los niveles de madurez.

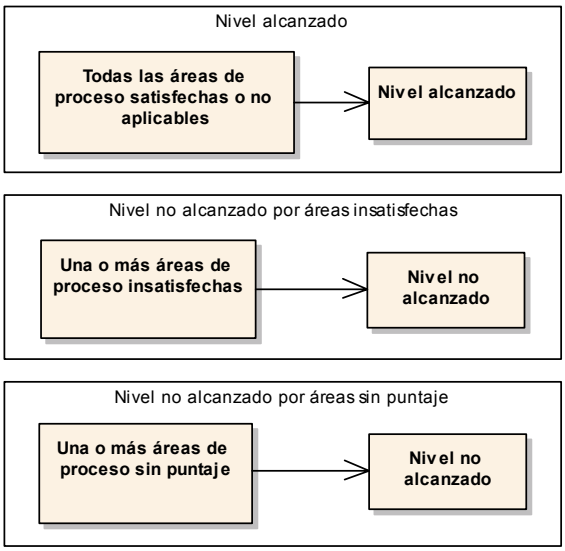

Figura 6.7. Reglas del negocio para la evaluación de los niveles de madurez.

Artefactos directos

public «entity» Object: Son los elementos tangibles que resultan directamente de la implementación de una práctica del modelo CMMI-SW.

Ejemplos: documentos, entregables, materiales de entrenamiento, etc.

Artefactos indirectos

public «entity» Object: Son elementos que surgen como consecuencia de la implementación de una práctica del modelo CMMI-SW, pero que no constituyen en sí el propósito de la práctica.

Ejemplos: minutas de reunión, revisión de resultados, reportes de estado, mediciones de desempeño, etc.

Observaciones

public «entity» Object: Son elementos que se recopilan durante el análisis de una organización o proyecto. Las observaciones resaltan las fortalezas y debilidades encontradas en la organización con respecto a la implementación de las prácticas de CMMI-SW. Generalmente se utilizan para documentar las debilidades encontradas.

Valoraciones de objetivos

public «entity» Object: A cada objetivo se le asigna una valoración de acuerdo al método SCAMPI.
Las valoraciones posibles son:

- Satisfecho
- Insatisfecho

Valoraciones de prácticas

public «entity» Object: A cada práctica se le asigna una valoración de acuerdo al método SCAMPI.
Las valoraciones posibles son:

- CI: completamente implementada
- AI: ampliamente implementada
- PI: parcialmente implementada
- NI: no implementada

Valoraciones de áreas de proceso

public «entity» Object: A cada área de proceso se le asigna una valoración de acuerdo al método SCAMPI.
Las valoraciones posibles son:

- Satisfecha
- Insatisfecha
- No aplicable
- Sin puntaje

Valoración del nivel de madurez

public «entity» Object: Es el resultado final del proceso de evaluación.
Las valoraciones posibles son:

- Nivel alcanzado
- Nivel no alcanzado

6.3 Requisitos del sistema

Los requisitos del sistema se identificaron durante el proceso EVS (Estudio de Viabilidad del Sistema) y se encuentran detallados en el Capítulo 4.

6.4 Glosario de términos

El glosario de términos utilizados en el dominio de aplicación se encuentra detallado en el apartado 13.2 (Glosario de términos correspondientes al análisis) del Capítulo 13.

6.5 Entorno tecnológico del sistema

El sistema se planteará como una herramienta de software monousuario, ejecutable en computadoras personales y con amplias capacidades gráficas de interfaz con el usuario. No poseerá componentes distribuidos ni necesitará infraestructura de red alguna. Teniendo en cuenta estas restricciones, la plataforma tecnológica sobre la cual se construirá la herramienta será J2SE (Edición Estándar de Java versión 2, en inglés *Java 2 Standard Edition*) [J2SE, 2004].

Esta plataforma permitirá que la herramienta obtenida pueda ejecutarse sin cambios bajo distintos sistemas operativos, entre los cuales se pueden citar todas las variantes de Windows, Linux, HP-UX, Solaris, y FreeBSD. Además de las facilidades multiplataforma, el lenguaje Java y la plataforma J2SE cuentan con amplias facilidades de construcción de interfaces gráficas de usuario, y con un gran nivel de madurez y aceptación dentro de la industria del software.

De acuerdo a lo expuesto anteriormente, el entorno tecnológico necesario para el sistema es el siguiente:

- Computadora Personal Pentium II o superior, con al menos 64 MB de memoria RAM.
- Ambiente de ejecución y/o desarrollo de la plataforma J2SE, más conocido como JDK (Java Development Kit) o JRE (Java Runtime Environment), versión 1.4.2 o superior [J2SE, 2004].

6.6 Catálogo de involucrados

La tabla 6.1 muestra los principales involucrados en el desarrollo de la herramienta. Cada involucrado se presenta bajo un rol con determinadas responsabilidades. Finalmente, se muestran las personas físicas que cubrirán los roles definidos.

Rol	*Responsabilidades*	*Representantes*
Usuario	Ayudar a entender el problema a resolver y a definir los requisitos del sistema. Validar los requisitos del sistema, las interfaces de usuario, y las funcionalidades en general. Verificar el correcto funcionamiento del sistema una vez finalizado.	M. Ing. Paola Britos Ing. Mario L. Peralta
Tesista	Desarrollar la herramienta siguiendo paso a paso la metodología Métrica V3, definiendo la herramienta con el usuario y planificando el desarrollo con el director de tesis. Documentar todos los productos obtenidos durante el desarrollo. Escribir el documento de tesis.	Ing. Mario L. Peralta
Director	Controlar el desarrollo de la herramienta de acuerdo a la metodología Métrica V3. Definir la planificación conjuntamente con el tesista. Supervisar los productos obtenidos durante el desarrollo y el documento de tesis.	M. Ing. Paola Britos
Co-Director	Supervisar los contenidos del documento de tesis Supervisar el funcionamiento de la herramienta	M. Ing. Eduardo Diez

Tabla 6.1. Principales involucrados en el desarrollo.

6.7 Modelo de casos de uso

A continuación se muestra un reporte del modelo de casos de uso generado en *Enterprise Architect*.

Reporte: Modelo de casos de uso

El diagrama de la figura 6.8 muestra los actores y casos de uso del sistema.

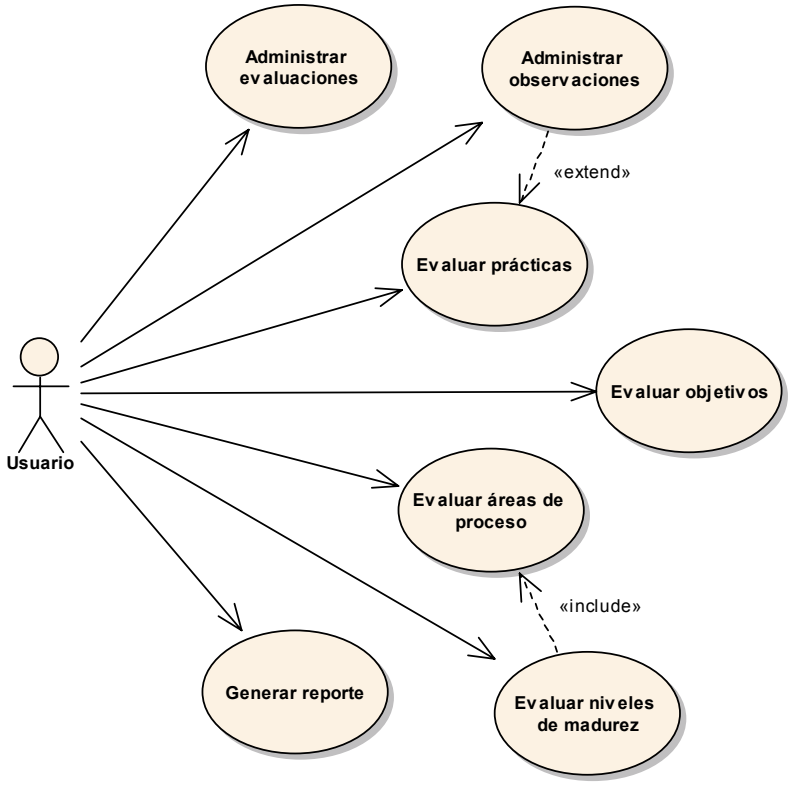

Figura 6.8. Actores y Casos de uso del sistema.

Modelo de casos de uso

public Package: El Modelo de casos de uso contiene los Actores y Casos de uso del sistema. Las relaciones entre los Actores y los Casos de uso se muestran en diagramas de casos de uso.

El modelo contiene además las especificaciones detalladas de cada uno de los casos de uso del sistema.

Usuario

public Actor: Representa al usuario (experto o novato) que utiliza el sistema como herramienta de asistencia para la evaluación de una organización o proyecto de software.

Administrar evaluaciones

public Use Case: Brinda la posibilidad de iniciar evaluaciones, de almacenar en un archivo los resultados de una evaluación completa o en curso, y de recuperar una evaluación almacenada en archivo para su modificación.

Para la selección de los archivos, el sistema provee una interfaz gráfica que permite navegar en el sistema de archivos.

Scenarios

Precondiciones {Property}.
Para el Flujo básico: Iniciar evaluación, no hace falta ninguna precondición.
Para el Flujo alternativo: Almacenar evaluación, debe existir una evaluación en curso.
Para el Flujo alternativo: Recuperar evaluación, debe existir una evaluación almacenada en archivo.

Postcondiciones {Property}.
Para el Flujo básico: Iniciar evaluación, el sistema ha iniciado una evaluación en blanco donde el usuario podrá ir completando todas las valoraciones.
Para el Flujo alternativo: Almacenar evaluación, la evaluación en curso queda almacenada en el archivo indicado por el usuario.
Para el Flujo alternativo: Recuperar evaluación, la evaluación almacenada en el archivo queda cargada en el sistema para que el usuario la complete o modifique.

Flujo básico: Iniciar evaluación {Basic Path}.

1. El sistema presenta al usuario una interfaz para que indique el nombre de la organización o proyecto a evaluar y el alcance de la evaluación (áreas de proceso, objetivos o prácticas). Para el alcance a nivel de prácticas habilita la posibilidad de definir un conjunto de instancias de evaluación.
2. El usuario provee los datos de organización y de alcance.
3. El sistema inicia una nueva evaluación.

Alternativa:
1. En caso de existir una evaluación en curso, el sistema presenta al usuario la oportunidad de utilizar el Flujo alternativo: Almacenar evaluación para almacenarla.

Flujo alternativo: Almacenar evaluación {Alternate}.

1. El sistema presenta al usuario una interfaz para navegar el sistema de archivos de su computadora.
2. El usuario indica el directorio y archivo donde almacenará la evaluación.
3. El sistema almacena la evaluación en el archivo indicado.

Flujo alternativo: Recuperar evaluación {Alternate}.

1. El sistema presenta al usuario una interfaz para navegar el sistema de archivos de su computadora.
2. El usuario indica el directorio y archivo donde se encuentra almacenada la evaluación.

3. El sistema recupera la evaluación desde el archivo indicado.

Alternativa:
1. En caso de existir una evaluación en curso, el sistema presenta al usuario la oportunidad de utilizar el Flujo alternativo: Almacenar evaluación para almacenarla.

Administrar observaciones

public Use Case: Brinda la posibilidad de crear nuevas observaciones, y de modificar o eliminar observaciones existentes. El usuario podrá vincular y desvincular estas observaciones con las prácticas del modelo.

Cada observación tiene un atributo que la caracteriza como debilidad o fortaleza y otro que indica si su impacto es significativo o no.

Scenarios

Precondiciones {Property}.
El usuario utilizó el caso de uso Administrar evaluaciones, subflujo Iniciar evaluación, especificando los detalles y el alcance de la evaluación a efectuar.

Postcondiciones {Property}.
En el caso de creación de observaciones, se tienen las nuevas observaciones disponibles para ser vinculadas a las prácticas.
En el caso de modificación o eliminación de observaciones, se recalculan los valores sugeridos para todas las prácticas que tuvieran vinculaciones con las observaciones modificadas o eliminadas. Además, el sistema muestra al usuario las valoraciones recalculadas de manera que el mismo pueda revisarlas nuevamente.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Observaciones".

Flujo básico {Basic Path}.
1. El sistema muestra una lista con las observaciones existentes. La lista contiene un número de observación, una descripción, el tipo (fortaleza o debilidad), y el impacto (significativo o no significativo).
2. El usuario puede seleccionar una observación de la lista y modificarla (Flujo alternativo: Modificar observación) o eliminarla (Flujo alternativo: Eliminar observación). Asimismo, sin seleccionar ninguna observación de la lista puede crear una nueva observación (Flujo alternativo: Crear observación).

Flujo alternativo: Crear observación {Alternate}.

1. El sistema presenta al usuario una interfaz para que indique el tipo (debilidad o fortaleza), el impacto (significactivo o no significativo) y la descripción de la observación.
2. El usuario provee los datos (tipo, impacto y descripción).
3. El sistema crea una nueva observación asignándole automáticamente un número secuencial que la identifique.

Flujo alternativo: Modificar observación {Alternate}.

1. El sistema presenta al usuario una interfaz con los valores actuales de la observación seleccionada (tipo, impacto, descripción).
2. El usuario modifica los datos que desea.
3. El sistema almacena la observación.

Alternativa:

3. Si el usuario modificó los datos tipo o impacto, el sistema recalcula las valoraciones sugeridas para todas las prácticas que tengan vinculaciones con la observación modificada, y presenta al usuario una lista de las prácticas recalculadas de manera que el mismo pueda volver a evaluarlas.

Flujo alternativo: Eliminar observación {Alternate}.

1. El sistema pregunta al usuario si está seguro de eliminar la observación seleccionada.
2. El usuario confirma la eliminación.
3. El sistema elimina la observación, recalcula las valoraciones sugeridas para todas las prácticas que tengan vinculaciones con la observación eliminada, y presenta al usuario una lista de las prácticas recalculadas de manera que el mismo pueda volver a evaluarlas.

Alternativa:

3. En caso de que el usuario no confirme la eliminación, el sistema permanece inalterado.

Evaluar niveles de madurez

public Use Case: Brinda la posibilidad de evaluar el nivel de madurez del modelo CMMI-SW alcanzado.

El usuario selecciona el nivel de madurez que desea evaluar y el sistema le indica las áreas de proceso asociadas. El usuario las evalúa utilizando el caso de uso Evaluar áreas de proceso y finalmente el sistema le responde si el nivel es alcanzado o no.

El usuario puede consultar en las guías online del sistema toda la información relativa al nivel de madurez que está analizando (criterios para su evaluación, áreas de proceso asociadas, etc).

Scenarios

Precondiciones {Property}.
El usuario utilizó el caso de uso Administrar evaluaciones, subflujo Iniciar evaluación, especificando los detalles y el alcance de la evaluación a efectuar.

Postcondiciones {Property}.
En caso de éxito, se obtiene como resultado una valoración para el nivel de madurez evaluado. En caso de fracaso o cancelación, el estado del sistema permanece inalterado.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Evaluar" al estar posicionado sobre un nivel de madurez del modelo.

Flujo básico {Basic Path}.
1. El sistema despliega las áreas de proceso asociadas al nivel de madurez evaluado.
2. El usuario utiliza el caso de uso Evaluar áreas de proceso para evaluar cada una de las áreas de proceso desplegadas.
3. El sistema muestra la valoración concluída para el nivel de madurez, basada en las valoraciones de las áreas de proceso asociadas al mismo (ver Criterios de evaluación).

Excepciones al Flujo básico {Alternate}.
En el paso 1 del flujo básico, para el caso de los niveles 3 a 5 del modelo, si no se encuentra evaluado con Valoración = Alcanzado el nivel anterior, el sistema muestra un mensaje de advertencia: "Antes de evaluar el Nivel de madurez N, deben haberse alcanzado todos los niveles inferiores", donde N se reemplaza por los valores 3 a 5.

El usuario puede aceptar el mensaje y seguir adelante con la evaluación.

Flujo alternativo: Evaluación de Nivel 3 {Alternate}.
Cuando se evalúa el Nivel 3 del modelo, la evaluación es a nivel de Prácticas, y se encuentra evaluado con Valoración = Alcanzado el nivel 2, antes del paso 1 del flujo básico el sistema agrega prácticas a los Objetivos Genéricos del Nivel 2. Esto hace que se vuelva a abrir la evaluación del nivel 2.

Criterios de evaluación {Property}.

Los criterios de evaluación de los niveles de madurez a partir de las valoraciones de las áreas de proceso son los siguientes:

- Si Todas las Áreas de proceso = Satisfecha o No aplicable, entonces Valoración de nivel de madurez = Alcanzado
- Si Una o más Áreas de proceso = Insatisfecha, entonces Valoración de nivel de madurez = No alcanzado
- Si Una o más Áreas de proceso = Sin puntaje, entonces Valoración de nivel de madurez = No alcanzado

Evaluar áreas de proceso

public Use Case: Brinda la posibilidad de evaluar el nivel de satisfacción de cada una de las áreas de proceso asociadas a los niveles de madurez del modelo CMMI-SW.

Para cada área de proceso, el usuario puede asignar alguno de los valores "Satisfecho" o "No satisfecho", con un comentario que justifique su valoración.

Si el usuario utilizó previamente el caso de uso Evaluar objetivos para evaluar los objetivos del área de proceso bajo análisis, el sistema sugerirá los valores "Satisfecho" o "No satisfecho" de acuerdo a la combinación de los valores asignados a las objetivos.

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

El usuario puede consultar en las guías online del sistema toda la información relativa al área de proceso que está analizando (criterios para su evaluación, objetivos asociados, etc).

Scenarios

Precondiciones {Property}.

El usuario utilizó el caso de uso Administrar evaluaciones, subflujo Iniciar evaluación, especificando los detalles y el alcance de la evaluación a efectuar.

Alternativas:

a) En el alcance indicó que la evaluación llegaría hasta el nivel de los objetivos o hasta el nivel de las prácticas. A continuación, evaluó todos los objetivos que forman parte del área de proceso.
b) En el alcance indicó que la evaluación llegaría hasta el nivel de las áreas de proceso. A continuación, seleccionó un nivel de madurez y un área de proceso.

Postcondiciones {Property}.
En caso de éxito, se obtiene como resultado una valoración para el área de proceso evaluada. En caso de fracaso o cancelación, el estado del sistema permanece inalterado.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Evaluar" al estar posicionado sobre un área de proceso del modelo.

Flujo básico {Basic Path}.
1. El sistema muestra una interfaz donde expone al usuario los criterios que debe tener en cuenta para evaluar el área de proceso seleccionada, junto con una opción para indicar que el área de proceso está fuera de alcance, otra opción para indicar que no hay evidencia objetiva, una lista de valoraciones posibles y un área para justificar su decisión. El sistema muestra además una valoración sugerida para el área de proceso, basada en las valoraciones de los objetivos que forman parte de la misma (ver Criterios de evaluación).
2. El usuario adopta la valoración sugerida por el sistema o indica su propia valoración justificando su decisión.
3. El sistema almacena la valoración indicada por el usuario.

Excepciones al flujo básico {Alternate}.
Si el alcance de la evaluación llega hasta el nivel de los objetivos o de las prácticas (precondición a) y el usuario aún no ha evaluado todos los objetivos que forman parte del área de proceso, en el paso 1 el sistema le responde con un mensaje de advertencia: "Antes de evaluar un Área de proceso debe evaluar todos los Objetivos de la misma".
El usuario puede aceptar el mensaje y seguir adelante con la evaluación. El sistema le muestra la interfaz de evaluación pero no le permite asignar una valoración.
Si el usuario selecciona las opciones "Fuera de alcance" o "No hay evidencia objetiva", el sistema sugiere una valoración de acuerdo a los Criterios de evaluación y habilita la posibilidad de asignar una valoración.

Flujo alternativo: Alcance hasta nivel de áreas de proceso {Alternate}.
Si en la inicialización de la evaluación se definió que el alcance llegaría hasta el nivel de las áreas de proceso, en el paso 1 del Flujo básico el sistema no presenta ninguna valoración sugerida, quedando a cargo del usuario asignar una valoración para el área de proceso.

Criterios de evaluación {Property}.
Los criterios de evaluación de las áreas de proceso a partir de las valoraciones de los objetivos son los siguientes:

- Si Todos los Objetivos genéricos = Satisfecho y Todos los Objetivos específicos = Satisfecho, entonces Valoración de área de proceso = Satisfecha
- Si Uno o más Objetivos genéricos = Insatisfecho, entonces Valoración de área de proceso = Insatisfecha
- Si Uno o más Objetivos específicos = Insatisfecho, entonces Valoración de área de proceso = Insatisfecha
- Si Área de proceso = Fuera de alcance, entonces Valoración de área de proceso = No aplicable
- Si Área de proceso = Sin evidencia objetiva, entonces Valoración de área de proceso = Sin puntaje

Evaluar objetivos

public Use Case: Brinda la posibilidad de evaluar el nivel de satisfacción de cada uno de los objetivos asociados a las áreas de proceso del modelo CMMI-SW.

Para cada objetivo, el usuario puede asignar alguno de los valores "Satisfecho" o "No satisfecho", con un comentario que justifique su valoración.

Si el usuario utilizó previamente el caso de uso Evaluar prácticas para evaluar las prácticas asociadas al objetivo bajo análisis, el sistema sugerirá los valores "Satisfecho" o "No satisfecho" de acuerdo a la combinación de los valores asignados a las prácticas.

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

El usuario puede consultar en las guías online del sistema toda la información relativa al objetivo que está analizando (criterios para su evaluación, prácticas asociadas, etc).

Scenarios

Precondiciones {Property}.
El usuario utilizó el caso de uso Administrar evaluaciones, subflujo Iniciar evaluación, especificando los detalles y el alcance de la evaluación a efectuar.

Alternativas:
a) En el alcance indicó que la evaluación llegaría hasta el nivel de las prácticas. A continuación, evaluó todas las prácticas que forman parte del objetivo.
b) En el alcance indicó que la evaluación llegaría hasta el nivel de los objetivos. A continuación, seleccionó un nivel de madurez, un área de proceso, y un objetivo.

Postcondiciones {Property}.
En caso de éxito, se obtiene como resultado una valoración para el objetivo evaluado. En caso de fracaso o cancelación, el estado del sistema permanece inalterado.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Evaluar" al estar posicionado sobre un objetivo del modelo.

Flujo básico {Basic Path}.
1. El sistema muestra una interfaz donde expone al usuario los criterios que debe tener en cuenta para evaluar el objetivo seleccionado, junto con una lista de valoraciones posibles y un área para justificar su decisión. El sistema muestra además una valoración sugerida para el objetivo, basada en las valoraciones de las prácticas que forman parte del mismo (ver Criterios de evaluación).
2. El usuario adopta la valoración sugerida por el sistema o indica su propia valoración justificando su decisión.
3. El sistema almacena la valoración indicada por el usuario.

Excepciones al flujo básico {Alternate}.
Si el alcance de la evaluación llega hasta el nivel de las prácticas (precondición a) y el usuario aún no ha evaluado todas las prácticas que forman parte del objetivo, en el paso 1 el sistema le responde con un mensaje de advertencia: "Antes de evaluar un Objetivo debe evaluar todas las Prácticas del mismo".

El usuario puede aceptar el mensaje y seguir adelante con la evaluación. El sistema le muestra la interfaz de evaluación pero no le permite asignar una valoración.

Flujo alternativo: Alcance hasta nivel de objetivos {Alternate}.
Si en la inicialización de la evaluación se definió que el alcance llegaría hasta el nivel de los objetivos, en el paso 1 del Flujo básico el sistema no presenta ninguna valoración sugerida, quedando a cargo del usuario asignar una valoración para el objetivo.

Criterios de evaluación {Property}.
Los criterios de evaluación de los objetivos a partir de las valoraciones de las prácticas son los siguientes:
- Si Prácticas = CI o AI y Cant. de Observaciones tipo debilidad de impacto significativo = 0, entonces Valoración de objetivo = Satisfecho

- Si Cant. de Observaciones tipo debilidad de impacto significativo > 0, entonces Valoración de objetivo = Insatisfecho
- Si una o más Prácticas = PI o NI, entonces Valoración de objetivo = Insatisfecho

Evaluar prácticas

public Use Case: Brinda la posibilidad de evaluar el nivel de implementación de cada una de las prácticas asociadas a los objetivos del modelo CMMI-SW.

El usuario indica la presencia de artefactos directos o indirectos que confirmen la implementación de la práctica, e indica si hay observaciones caracterizadas como debilidad asociadas con la práctica.

De acuerdo a la información ingresada por el usuario, el sistema sugerirá alguno de los siguientes valores:

- CI: completamente implementada
- AI: ampliamente implementada
- PI: parcialmente implementa
- NI: no implementada

El usuario puede sobreescribir los valores sugeridos por el sistema, justificando su decisión con un comentario.

Las prácticas podrán evaluarse a nivel de proyecto o a nivel de organización. En caso de evaluarse a nivel de proyecto para más de un proyecto, el sistema asignará los valores a nivel de organización de acuerdo a la combinación de los valores a nivel de proyecto siguiendo las reglas del método SCAMPI.

El usuario puede consultar en las guías online del sistema toda la información relativa a la práctica que está analizando (artefactos directos e indirectos, criterios para su evaluación, etc).

Scenarios

Precondiciones {Property}.

El usuario utilizó el caso de uso Administrar evaluaciones, subflujo Iniciar evaluación, especificando los detalles y el alcance de la evaluación a efectuar. En cuanto al alcance, indicó que la evaluación llegaría hasta el nivel de las prácticas.

A continuación, seleccionó un nivel de madurez, un área de proceso, un objetivo, y una práctica dentro del mismo.

Postcondiciones {Property}.

En caso de éxito, se obtiene como resultado una valoración para la práctica evaluada, ya sea en el contexto de una instancia o de un conjunto de instancias.

En caso de fracaso o cancelación, el estado del sistema permanece inalterado.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Evaluar" al estar posicionado sobre una práctica del modelo.

Flujo básico {Basic Path}.

1. El sistema muestra una interfaz donde expone al usuario los criterios que debe tener en cuenta para evaluar la práctica seleccionada y un conjunto de elementos para que el usuario provea la información necesaria.
2. Si en la inicialización de la evaluación (ver Administrar evaluaciones) se definieron múltiples instancias, el usuario podrá seleccionar una instancia o indicar que va a evaluar el conjunto de instancias. En caso de haber seleccionado una instancia, el usuario provee la información sobre artefactos directos, artefactos indirectos y observaciones vinculados a la práctica bajo análisis.
   Los valores posibles para cada elemento son los siguientes:
   - Artefactos directos: Apropiados, No apropiados o ausentes, No se sabe.
   - Artefactos indirectos: Confirman, Sugieren, No se sabe.
   - Observaciones: seleccionables de una lista de observaciones (administrada con el caso de uso Administrar observaciones)
3. El sistema muestra una valoración sugerida para la práctica basada en la información provista por el usuario (para evaluación a nivel de instancia) o en las valoraciones de instancia (para evaluación a nivel conjunto de instancias). Los criterios que sigue el sistema se detallan en Criterios de evaluación.
4. El usuario adopta la valoración sugerida por el sistema o indica su propia valoración justificando su decisión.
5. El sistema almacena la valoración indicada por el usuario.

Excepciones al Flujo básico {Alternate}.
En el paso 2, si el usuario indica que va a evaluar el conjunto de instancias y aún no ha evaluado todas las instancias particulares que forman el conjunto, el sistema le responde con un mensaje de advertencia: "Antes de evaluar una Práctica a nivel de conjunto debe evaluar todas las instancias del mismo".

El usuario puede aceptar el mensaje y seguir adelante con la evaluación. El sistema le muestra la interfaz de evaluación pero no le permite asignar una valoración.

Flujo alternativo: No se definieron múltiples instancias {Alternate}.
Si en la inicialización de la evaluación se definió una única instancia para la evaluación, en el paso 2 del Flujo básico el usuario carece de la posibilidad de

seleccionar instancias y el sistema asume que se evalúa la única instancia definida. El resto de los pasos permanece inalterado.

Criterios de evaluación {Property}.
Los criterios de evaluación para el nivel de instancia son los siguientes:

- Si Artefactos directos = Apropiados y Artefactos indirectos = Confirman y Cant. de Observaciones tipo debilidad = 0, entonces Valoración de práctica = CI (Completamente Implementada)
- Si Artefactos directos = Apropiados y Artefactos indirectos = Confirman y Cant. de Observaciones tipo debilidad > 0, entonces Valoración de práctica = AI (Ampliamente Implementada)
- Si Artefactos directos = No apropiados o ausentes y Artefactos indirectos = Sugieren y Cant. de Observaciones tipo debilidad > 0, entonces Valoración de práctica = PI (Parcialmente Implementada)
- Cualquier otra combinación de artefactos y observaciones, entonces Valoración de práctica = NI (No Implementada)

Los criterios de evaluación a nivel conjunto de instancias son los siguientes:

- Si todas las instancias = X (CI, AI, PI, NI), entonces el conjunto = X (CI, AI, PI, NI)
- Si todas las instancias = CI o AI, entonces el conjunto = AI
- Si una o más instancias = PI y ninguna instancia = NI, entonces el conjunto = AI o PI
- Si una o más instancias = NI, entonces el conjunto = NI, PI o AI

Generar reporte

public Use Case: Brinda la posibilidad de generar un reporte con los resultados de la evaluación. El reporte muestra las valoraciones sugeridas, las valoraciones elegidas, y las justificaciones provistas por el usuario, para cada uno de los componentes del modelo (Niveles de madurez, Áreas de proceso, Objetivos, Prácticas).

Scenarios

Precondiciones {Property}.
Debe existir una evaluación en curso.

Postcondiciones {Property}.
Se genera un reporte de la evaluación en curso.

Activación {Property}.
Este caso de uso se activa cuando el usuario selecciona la opción "Generar reporte"

Flujo básico {Basic Path}.

1. El sistema presenta al usuario una presentación preliminar del reporte generado, mostrando para cada componente del modelo la Valoración sugerida, la Valoración elegida y la Justificacion.

Los componentes a incluir en el reporte dependen del alcance de la evaluación.

Para evaluación a nivel de Áreas de proceso:
Nivel de madurez
- Área de proceso 1
- Área de proceso 2
- ...
- Área de proceso N

Para evaluación a nivel de Objetivos:
Nivel de madurez
- Área de proceso 1
  - Objetivo específico 1
  - ...
  - Objetivo genérico
- Área de proceso 2
  - Objetivo específico 1
  - ...

Para evaluación a nivel de prácticas:
Nivel de madurez
- Área de proceso 1
  - Objetivo específico 1
    - Práctica 1
    - Práctica 2
    - ...
  - Objetivo genérico
    - Práctica 1
    - Práctica 2
    - ...
- Área de proceso 2
  - Objetivo específico 1
    - Práctica 1
    - Práctica 2
    - ...

2. El usuario puede imprimir el reporte si lo desea.

6.8 Especificación de interfaz de usuario

6.8.1 Principios generales de la interfaz

La interfaz con el usuario es gráfica e interactiva, del tipo estándar utilizado en todas las aplicaciones basadas en ventanas.

A continuación se muestran los lineamientos principales de la interfaz de usuario:

- La activación de las distintas operaciones de la herramienta se produce mediante opciones de una barra de menú y menúes contextuales.
- La aplicación muestra la estructura del modelo CMMI-SW mediante un árbol, donde los diferentes nodos representan los Niveles de madurez, las Áreas de proceso, los Objetivos y las Prácticas.
- El usuario puede seleccionar cualquier nodo del árbol que representa al modelo CMMI-SW y activar operaciones mediante un menú contextual (por ejemplo la operación Evaluar).
- La evaluación de los distintos componentes del modelo por parte del usuario se efectúa mediante formularios que contienen campos de texto, listas de selección simple o múltiple, botones de selección simple o múltiple, etc.
- Todos los formularios poseen un botón para aceptar los datos provistos por el usuario y otro para cancelarlos (botones Aceptar y Cancelar), y un botón para obtener ayuda (botón Ayuda). Todos los botones se ubican en la parte inferior del formulario.
- Todos los formularios de evaluación están divididos en dos partes. Una parte superior que contiene las guías online para la determinación de las valoraciones, y una parte inferior que contiene los campos editables por el usuario para el ingreso de datos.
- Los campos para mostrar valoraciones sugeridas y para ingresar valoraciones por parte del usuario se ubican en la parte inferior de los formularios de evaluación, por encima de los botones Aceptar, Cancelar y Ayuda.
- Los mensajes de error se muestran mediante ventanas emergentes.
- Las ayudas online asociadas a la herramienta se muestran en una ventana separada (ventana de Ayuda).
- En todos los formularios, menúes y ventanas de la herramienta, sólo estarán activadas las opciones que pueden ser utilizadas por el usuario. Aquellas opciones que no pueden ser utilizadas en un momento determinado, se encontrarán deshabilitadas.

6.8.2 Modelo de navegación de interfaz

En este modelo se contemplan las interfaces de usuario que existen en el sistema y la forma en que las mismas pueden navegarse.

A continuación se muestra un reporte de este modelo generado en *Enterprise Architect*.

Reporte: Modelo de navegación de interfaz

El diagrama de la figura 6.9 muestra las distintas interfaces del sistema y la forma en que se vinculan en ellas.

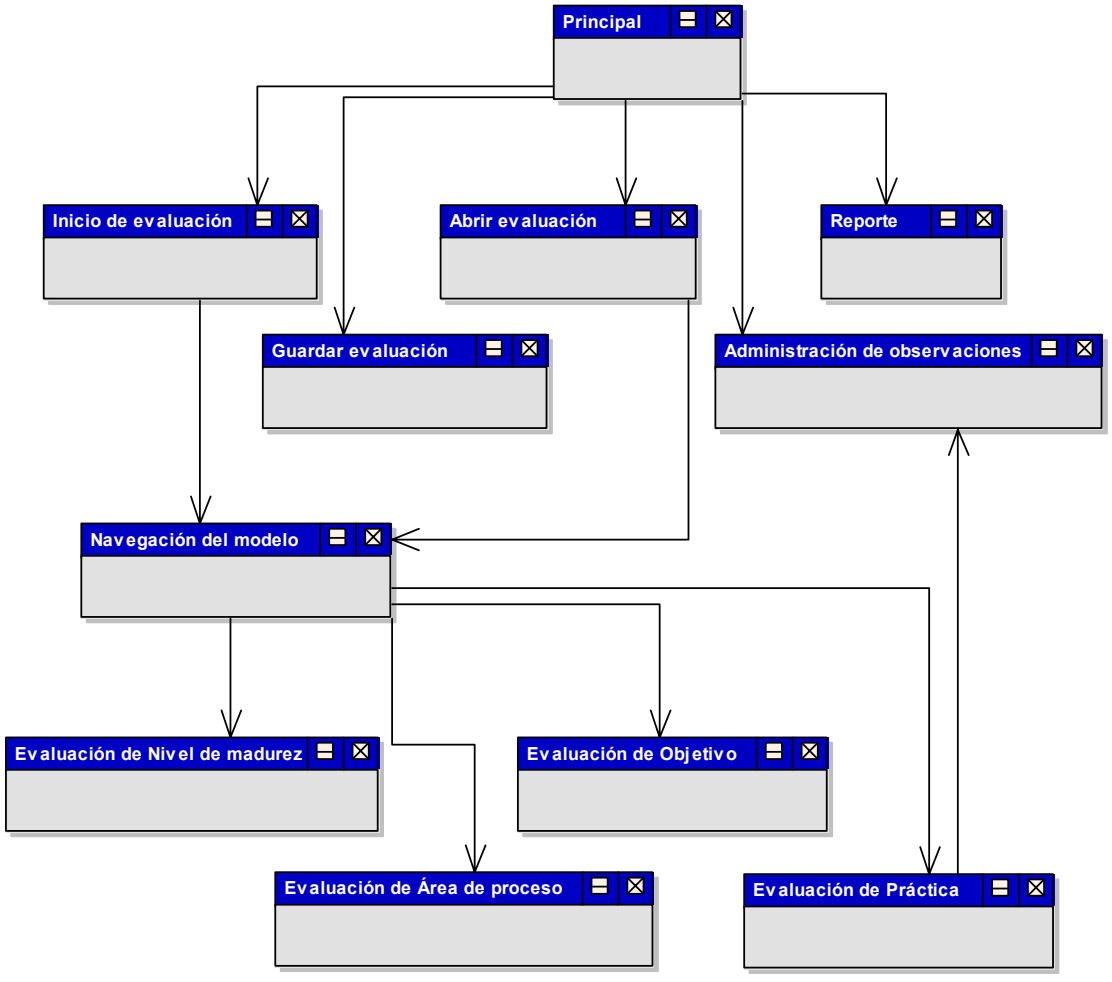

Figura 6.9. Modelo de navegación de interfaz.

Principal

public Screen: Es la ventana principal del sistema, madre de todas las demás interfaces. Contiene una barra de menú para que el usuario seleccione las operaciones que desea realizar.

Inicio de evaluación

public Screen: En esta ventana el usuario puede ingresar los datos de la organización o proyecto a evaluar y el alcance de la evaluación.

Abrir evaluación

public Screen: En esta ventana el usuario puede seleccionar un archivo de evaluación para cargarla dentro del sistema.

Guardar evaluación

public Screen: En esta ventana, el usuario puede indicar un archivo donde se va a almacenar la evaluación en curso.

Reporte

public Screen: En esta ventana el usuario puede visualizar e imprimir el reporte de la evaluación

Navegación del modelo

public Screen: Esta ventana muestra un árbol con la estructura del modelo CMMI-SW donde cada nodo representa un componente del mismo (Niveles de madurez, Áreas de proceso, Objetivos, Prácticas). El usuario puede seleccionar un componente y activar las distintas ventanas de evaluación.

Evaluación de Nivel de madurez

public Screen: En esta ventana el usuario puede evaluar un nivel de madurez determinado. La ventana muestra una guía online para la valoración y un formulario para el ingreso de datos.

Evaluación de Área de proceso

public Screen: En esta ventana el usuario puede evaluar un área de proceso determinada. La ventana muestra una guía online para la valoración y un formulario para el ingreso de datos.

Evaluación de Objetivo

public Screen: En esta ventana el usuario puede evaluar un objetivo determinado. La ventana muestra una guía online para la valoración y un formulario para el ingreso de datos.

Evaluación de Práctica

public Screen: En esta ventana el usuario puede evaluar una práctica específica del modelo. La ventana muestra una guía online para la valoración y un formulario para el ingreso de datos.

Administración de observaciones

public Screen: En esta ventana el usuario puede crear, modificar o eliminar Observaciones.

6.8.3 Prototipos de interfaz

El siguiente reporte de *Enterprise Architect* muestra los prototipos de cada una de las interfaces que componen la aplicación, las cuales fueron presentadas en el apartado anterior.

Reporte: Prototipos de interfaz

Principal

public Screen: La interfaz Principal presenta una barra de menú en su parte superior y un panel vacío en su parte inferior.

Luego del inicio de una nueva evaluación o de la apertura de una evaluación existente, en la parte izquierda del panel inferior se ubica la interfaz Navegación del modelo.

La figura 6.10 muestra el aspecto de esta interfaz.

Figura 6.10. Interfaz Principal

Inicio de evaluación

public Screen: La ventana presenta un área de texto para ingresar el nombre de la organización o proyecto a evaluar y una lista seleccionable para indicar el alcance de la evaluación.

Cuando se especifica un alcance al nivel de las Prácticas, se habilita la parte inferior de la ventana, donde se pueden definir un conjunto de instancias (o proyectos) de evaluación.

La figura 6.11 muestra el aspecto de esta interfaz.

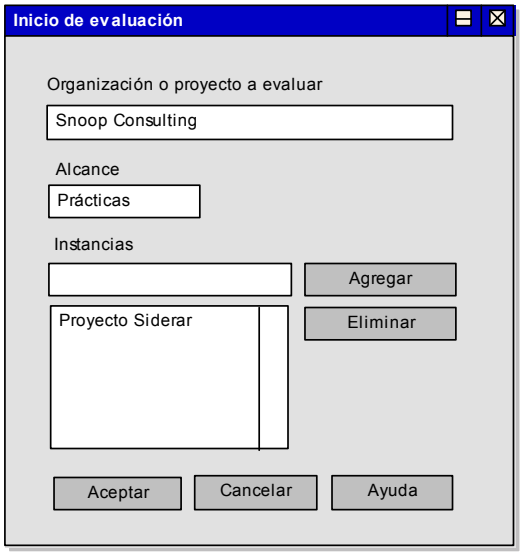

Figura 6.11. Inicio de evaluación.

Abrir/Guardar evaluación

public Screen: La ventana para las operaciones Abrir y Guardar es básicamente la misma. Por tal motivo, se realiza un único prototipo para ambas operaciones.

La ventana presenta en su parte superior una lista donde el usuario puede seleccionar unidades o directorios. Los contenidos de las unidades o directorios seleccionados (archivos y directorios) se muestran en el área central.

En la parte inferior hay un área para ingresar el nombre del archivo de destino, y una lista seleccionable de tipos de archivo en su parte inferior. Debajo de ellos, hay un botón para confirmar la operación (Abrir o Guardar) y otro para Cancelar.

La figura 6.12 muestra el aspecto de esta interfaz.

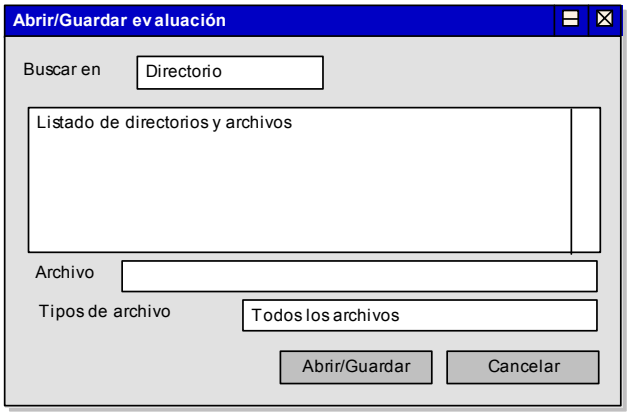

Figura 6.12. Abrir/Guardar evaluación.

Reporte

public Screen: La interfaz de reporte presenta un área con la presentación preliminar del reporte y un botón para la impresión del mismo. Asimismo, presenta botones para avanzar y retroceder páginas.

La figura 6.13 muestra el aspecto de esta interfaz.

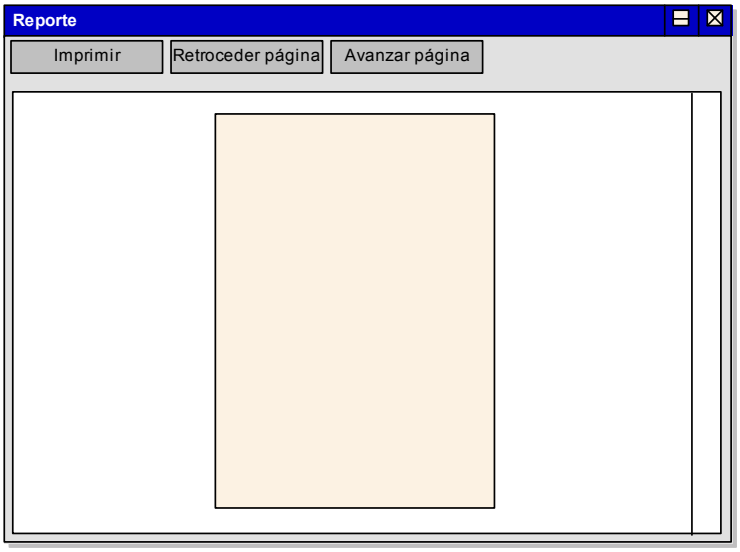

Figura 6.13. Reporte.

Navegación del modelo

public Screen: La ventana de navegación del modelo se encuentra embebida en la ventana principal, en la parte lateral izquierda. En ella el usuario puede ir desplegando los distintos niveles para ver las áreas de proceso, y dentro de ellas los objetivos y las prácticas.

En cualquier momento, el usuario podrá seleccionar un elemento (área de proceso, objetivo, práctica) y evaluarlo, ante lo cual se abrirá en la parte de la derecha la ventana de evaluación correspondiente.

La figura 6.14 muestra el aspecto de esta interfaz.

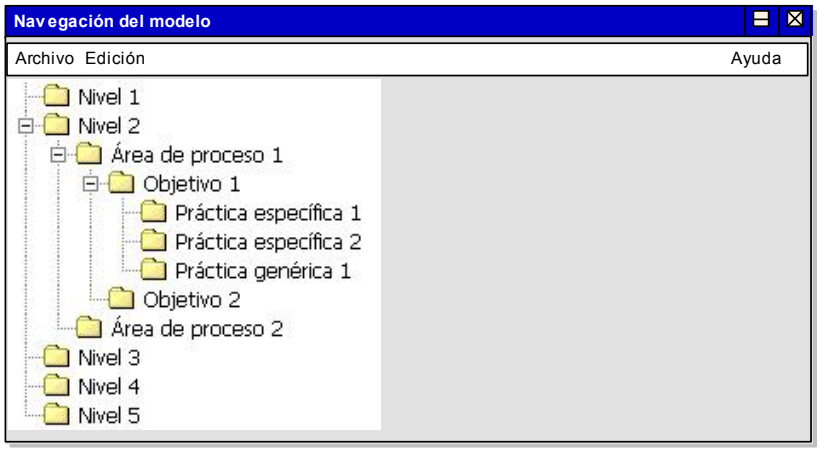

Figura 6.14. Navegación del modelo.

Evaluación de Nivel de madurez

public Screen: La ventana de evaluación de nivel de madurez se encuentra embebida en la parte lateral derecha de la ventana principal.

La misma presenta la guía para la valoración en su parte superior y la valoración concluída (en base a las valoraciones de las áreas de proceso) en su parte inferior. Debajo de la valoración concluída se ubica un botón para Aceptar que concluye la operación cerrando la ventana, y un botón para obtener Ayuda.

La figura 6.15 muestra el aspecto de esta interfaz.

Evaluación de Área de proceso

public Screen: La ventana de evaluación de área de proceso se encuentra embebida en la parte lateral derecha de la ventana principal.

La misma presenta la guía para la valoración en su parte superior, un área para el ingreso de datos en su parte central, y las valoraciones (sugerida y elegida) en su parte inferior. Debajo de las valoraciones se ubican los botones para Aceptar, Cancelar y obtener Ayuda.

La figura 6.16 muestra el aspecto de esta interfaz.

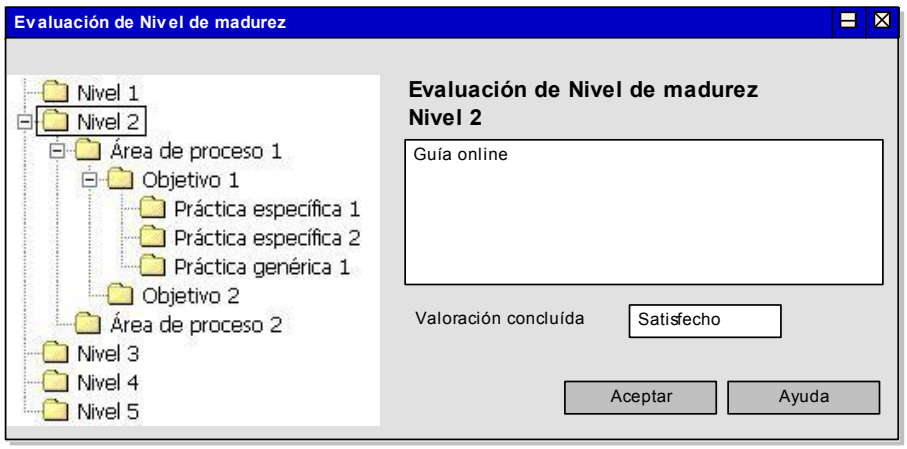

Figura 6.15. Evaluación de Nivel de madurez.

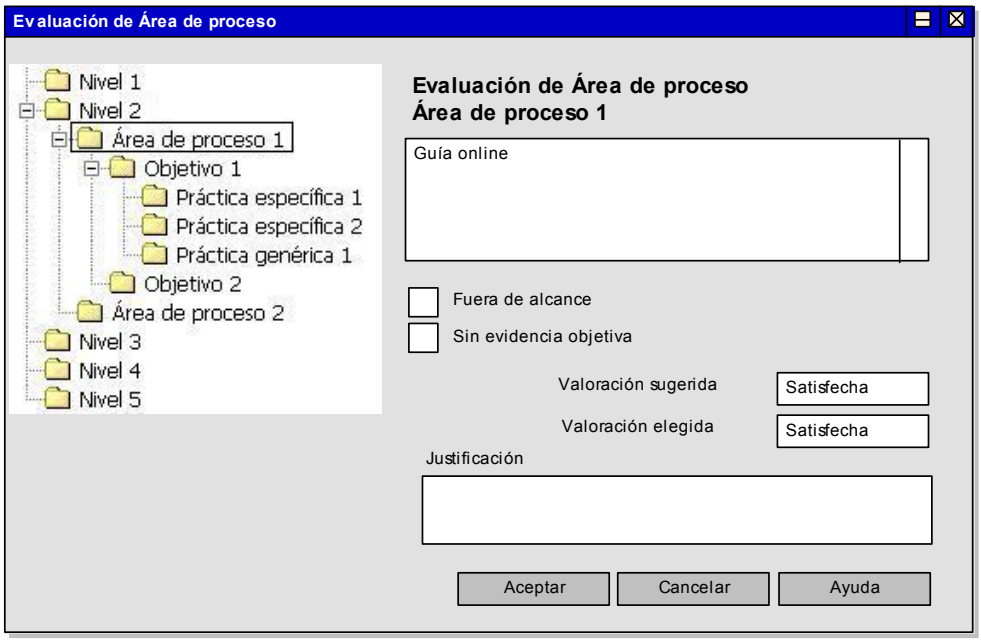

Figura 6.16. Evaluación de Área de proceso.

Evaluación de Objetivo

public Screen: La ventana de evaluación de objetivo se encuentra embebida en la parte lateral derecha de la ventana principal.

La misma presenta la guía para la valoración en su parte superior, y las valoraciones (sugerida y elegida) en su parte inferior. Debajo de las valoraciones se ubican los botones para Aceptar, Cancelar y obtener Ayuda.

La figura 6.17 muestra el aspecto de esta interfaz.

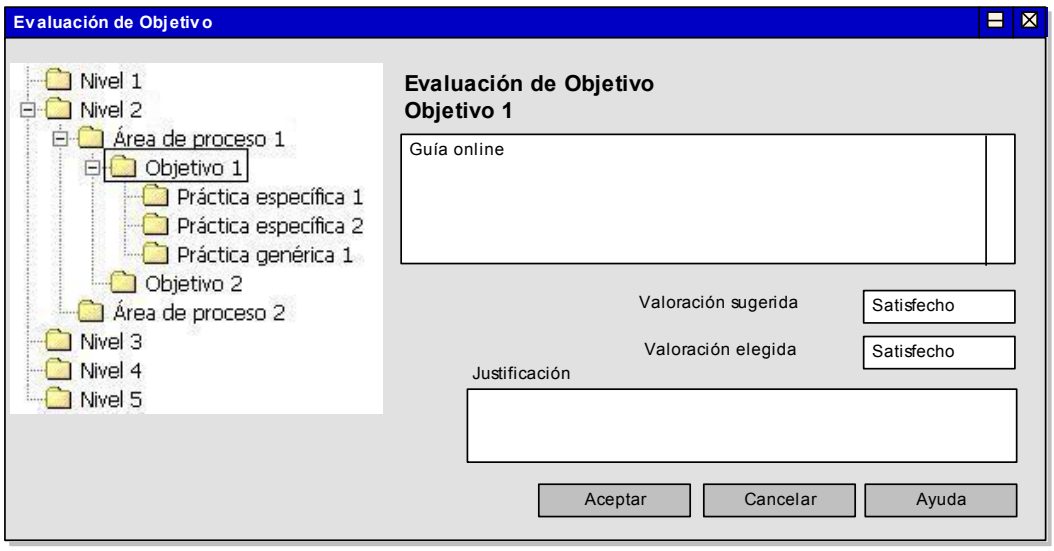

Figura 6.17. Evaluación de Objetivo.

Evaluación de Práctica

public Screen: La ventana de evaluación de práctica se encuentra embebida en la parte lateral derecha de la ventana principal.

La misma presenta la guía para la valoración en su parte superior, un área para el ingreso de datos en su parte central, y las valoraciones (sugerida y elegida) en su parte inferior. Debajo de las valoraciones se ubican los botones para Aceptar, Cancelar y obtener Ayuda. La figura 6.18 muestra el aspecto de esta interfaz.

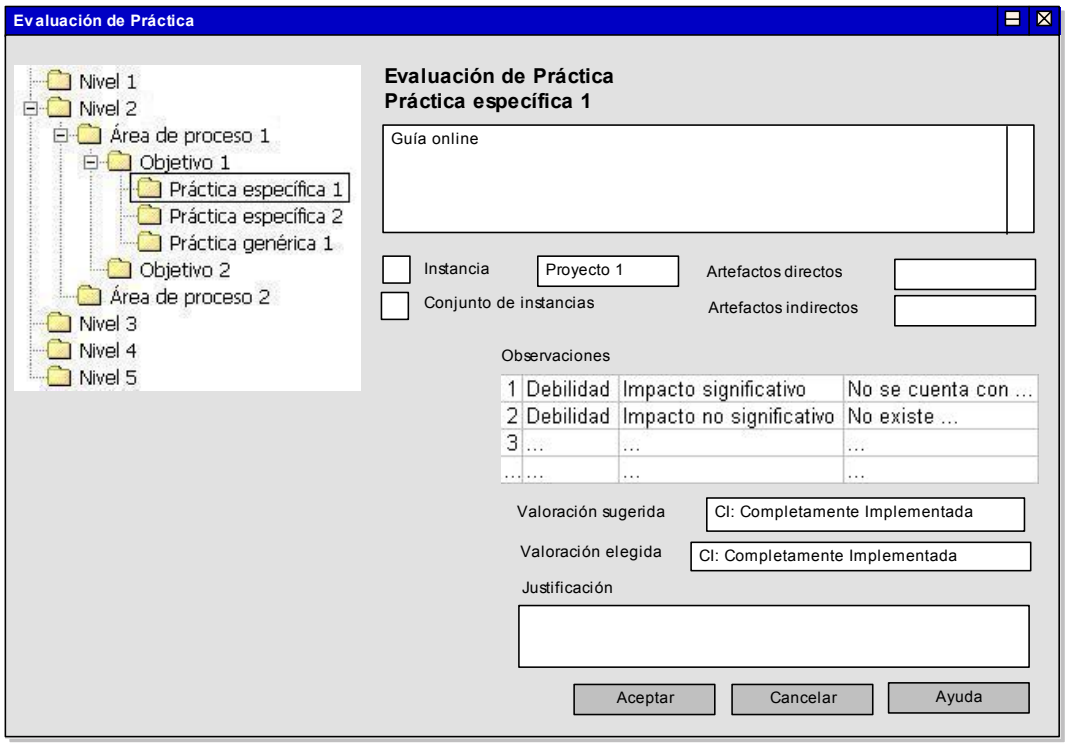

Figura 6.18. Evaluación de Práctica.

Administración de Observaciones

public Screen: La ventana presenta las observaciones existentes en la parte superior, y los campos para crear o modificar observaciones en la parte inferior. Debajo de estos campos se encuentran los botones para Agregar, Modificar y Eliminar observaciones; y los botones para Cerrar la ventana y obtener Ayuda.

Para agregar una nueva observación, el usuario ingresa los datos y presiona Agregar, agregándose la nueva observación al final de la lista. Para modificar, el usuario selecciona una observación de la lista, con lo que automáticamente se llenan los campos editables con los datos de la observación. El usuario modifica los datos que desea y presiona Modificar. Para eliminar, el procedimiento es el mismo que para modificar.

La figura 6.19 muestra el aspecto de esta interfaz.

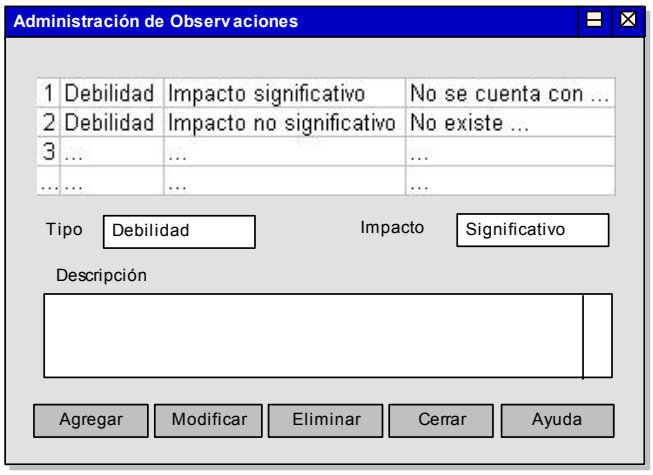

Figura 6.19. Administración de Observaciones.

6.9 Modelo de clases de análisis

El siguiente reporte de *Enterprise Architect* muestra el modelo de clases de análisis obtenido como resultado de las actividades "Análisis de los casos de uso" y "Análisis de clases" contempladas en la metodología.

Reporte: Modelo de clases de análisis

Este modelo contempla las clases de análisis identificadas a partir de los casos de uso. Las clases de análisis son básicamente de alguno de los siguientes tipos:

- **Clases de Interfaz <<boundary>>**: describen la interacción entre el sistema y sus actores. Representan interfaces de usuario y/o interfaces con otros sistemas o disponsitivos.
- **Clases de Control <<control>>**: son responsables de la coordinación de las acciones dentro de los casos de uso.
- **Clases de Entidad <<entity>>**: representan la información manipulada en el caso de uso.

El diagrama de clases de la figura 6.20 muestra las clases identificadas.

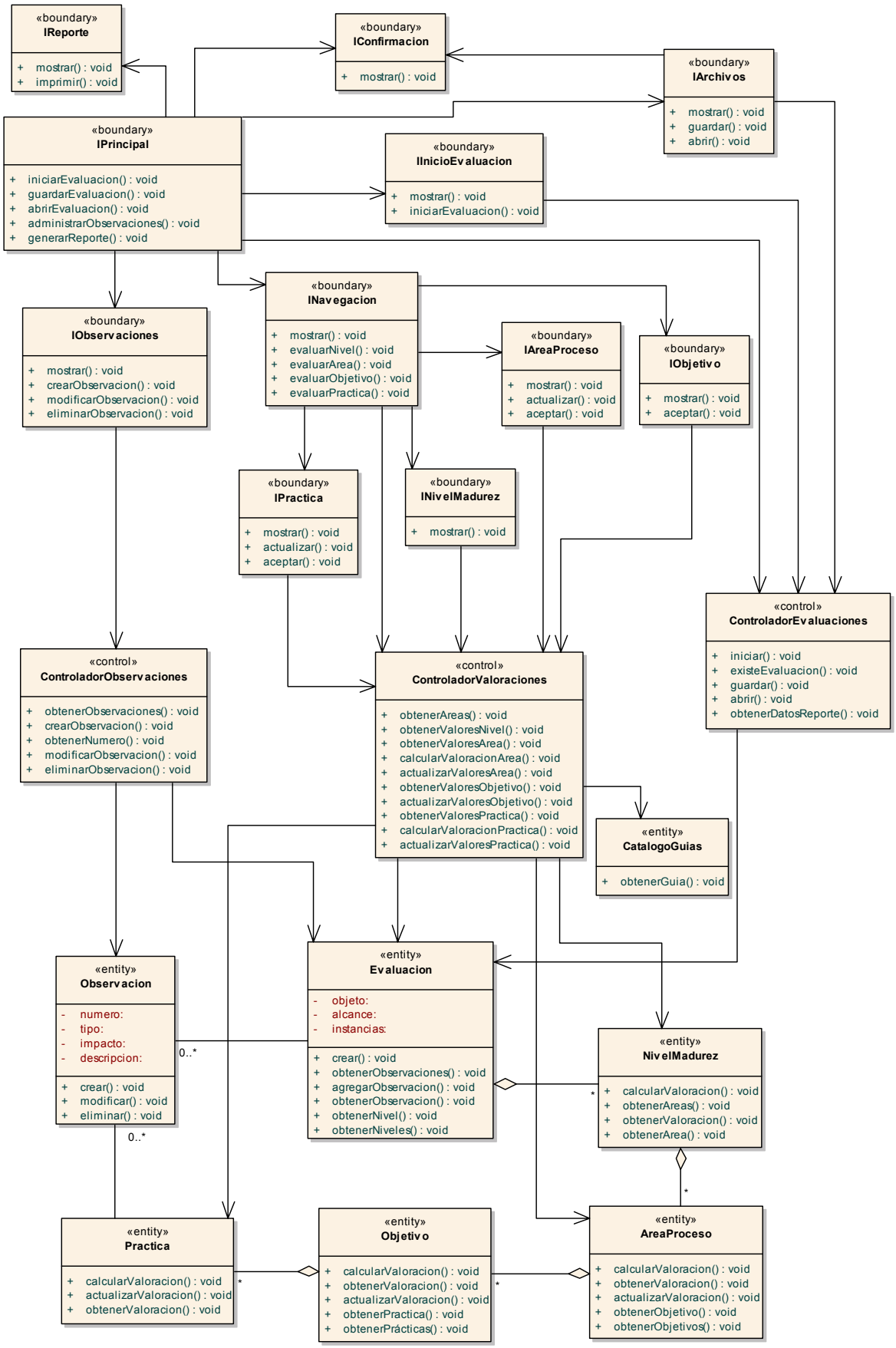

Figura 6.20. Clases de análisis.

IPrincipal

public «boundary» Class: Representa la interfaz Principal de la aplicación.

IPrincipal Methods

Method	Type	Notes
iniciarEvaluacion ()	public: *void*	Da la orden de iniciar una evaluación.
guardarEvaluacion ()	public: *void*	Da la orden de guardar una evaluación en archivo.
abrirEvaluacion ()	public: *void*	Da la orden de abrir una evaluación almacenada en archivo.
administrarObservaciones ()	public: *void*	Da la orden de mostrar la interfaz de administración de observaciones.
generarReporte ()	public: *void*	Da la orden de generar un reporte

IReporte

public «boundary» Class: Representa la ventana de previsualización e impresión de reportes.

IReporte Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
imprimir ()	public: *void*	Da la orden de imprimir el reporte mostrado en la interfaz

IConfirmacion

public «boundary» Class: Representa la interfaz de confirmacion de operaciones.

IConfirmacion Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.

IInicioEvaluacion

public «boundary» Class: Representa la interfaz de Inicio de evaluación.

IInicioEvaluacion Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
iniciarEvaluacion ()	public: *void*	Da la orden de iniciar una evaluación.

INavegacion

public «boundary» Class: Representa la interfaz de Navegación del modelo.

INavegacion Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
evaluarNivel ()	public: *void*	Da la orden de evaluar un nivel de madurez.
evaluarArea ()	public: *void*	Da la orden de evaluar un área de proceso.
evaluarObjetivo ()	public: *void*	Da la orden de evaluar un objetivo.
evaluarPractica ()	public: *void*	Da la orden de evaluar una práctica.

IArchivos

public «boundary» Class: Representa la interfaz para Abrir o Guardar evaluaciones en archivos.

IArchivos Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
guardar ()	public: *void*	Da la orden de guardar la evaluación en archivo.
abrir ()	public: *void*	Da la orden de abrir una evaluación guardada en archivo.

IObservaciones

public «boundary» Class: Representa la interfaz de Administración de observaciones.

IObservaciones Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
crearObservacion ()	public: *void*	Da la orden de crear una nueva observación.
modificarObservacion ()	public: *void*	Da la orden de modificar una observación.
eliminarObservacion ()	public: *void*	Da la orden de eliminar una observación.

INivelMadurez

public «boundary» Class: Representa la interfaz de Evaluación de niveles de madurez.

INivelMadurez Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.

IAreaProceso

public «boundary» Class: Representa la interfaz de Evaluación de áreas de proceso.

IAreaProceso Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
actualizar ()	public: *void*	Da la orden de actualizar la valoración sugerida.
aceptar ()	public: *void*	Da la orden de aceptar los datos indicados para la valoración.

IObjetivo

public «boundary» Class: Representa la interfaz de Evaluación de objetivos.

IObjetivo Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
aceptar ()	public: *void*	Da la orden de aceptar los datos indicados para la valoración.

IPractica

public «boundary» Class: Representa a la interfaz de Evaluación de prácticas.

IPractica Methods

Method	Type	Notes
mostrar ()	public: *void*	Muestra la interfaz.
actualizar ()	public: *void*	Da la orden de actualizar la valoración sugerida.
aceptar ()	public: *void*	Da la orden de aceptar los datos indicados para la valoración.

ControladorEvaluaciones

public «control» Class: Clase encargada del control relacionado con la administración de evaluaciones.

ControladorEvaluaciones Methods

Method	Type	Notes
iniciar ()	public: *void*	Inicia una nueva evaluación.
existeEvaluacion ()	public: *void*	Verifica si existe una evaluación en curso.
guardar ()	public: *void*	Guarda una evaluación en archivo.
abrir ()	public: *void*	Abre una evaluación guardada en archivo.
obtenerDatosReporte ()	public: *void*	Obtiene datos para el reporte

ControladorObservaciones

public «control» Class: Clase encargada del control relacionado con la administración de observaciones.

ControladorObservaciones Methods

Method	Type	Notes
obtenerObservaciones ()	public: *void*	Obtiene una lista con todas las observaciones de la evaluación.

Method	Type	Notes
crearObservacion ()	public: *void*	Gestiona la creación de una nueva observación.
obtenerNumero ()	public: *void*	Genera un nuevo número de observación.
modificarObservacion ()	public: *void*	Gestiona la modificación de una observación existente.
eliminarObservacion ()	public: *void*	Gestiona la eliminación de una observación existente.

ControladorValoraciones

public «control» Class: Clase encargada del control relacionado con las valoraciones de los distintos elementos del modelo.

ControladorValoraciones Methods

Method	Type	Notes
obtenerAreas ()	public: *void*	Obtiene las áreas de proceso contenidas en un nivel de madurez.
obtenerValoresNivel ()	public: *void*	Obtiene las guías y las valoraciones asociadas a un nivel de madurez.
obtenerValoresArea ()	public: *void*	Obtiene las guías y las valoraciones asociadas a un area de proceso.
calcularValoracionArea ()	public: *void*	Calcula la valoración sugerida para un área.
actualizarValoresArea ()	public: *void*	Actualiza la valoración de un área de proceso.
obtenerValoresObjetivo ()	public: *void*	Obtiene las guías y las valoraciones asociadas a un objetivo.
actualizarValoresObjetivo ()	public: *void*	Actualiza la valoración de un objetivo.
obtenerValoresPractica ()	public: *void*	Obtiene las guías y las valoraciones asociadas a una práctica.
calcularValoracionPractica ()	public: *void*	Calcula la valoración sugerida para una práctica.
actualizarValoresPractica ()	public: *void*	Actualiza la valoración de una práctica.

Evaluacion

public «entity» Class: Representa una evaluación de CMMI-SW aplicada a una organización o proyecto.

Evaluacion Attributes

Attribute	Type	Notes
objeto	private :	Organizacion o Proyecto objeto de la evaluación.
alcance	private :	Alcance de la evaluación. Los valores posibles son: Áreas de proceso, Objetivos, Prácticas.
instancias	private :	Lista de nombres con las instancias (proyectos) de evaluación.

Evaluacion Methods

Method	Type	Notes
crear ()	public: *void*	Crea una evaluación.
obtenerObservaciones ()	public: *void*	Obtiene la lista de observ. asociadas a la evaluación.
agregarObservacion ()	public: *void*	Agrega una observación a la evaluación.
obtenerObservacion ()	public: *void*	Obtiene una observación determinada de las asociadas a la evaluación.
obtenerNivel ()	public: *void*	Obtiene un nivel de madurez determinado.
obtenerNiveles ()	public: *void*	Obtiene todos los niveles de madurez

NivelMadurez

public «entity» Class: Representa un nivel de madurez del modelo CMMI-SW.

NivelMadurez Methods

Method	Type	Notes
calcularValoracion ()	public: *void*	Calcula la valoración asociada al nivel de madurez.
obtenerAreas ()	public: *void*	Obtiene las áreas de proceso contenidas en el nivel de madurez.
obtenerValoracion ()	public: *void*	Obtiene la valoracion concluida para el nivel.
obtenerArea ()	public: *void*	Obtiene un área determinada de las que corresponden al nivel.

AreaProceso

public «entity» Class: Representa un área de proceso del modelo CMMI-SW.

AreaProceso Methods

Method	Type	Notes
calcularValoracion ()	public: *void*	Calcula la valoración asociada al área de proceso.
obtenerValoracion ()	public: *void*	Obtiene la valoración asignada al área de proceso.
actualizarValoracion ()	public: *void*	Actualiza la valoración del área.
obtenerObjetivo ()	public: *void*	Obtiene un objetivo determinado de los pertenecientes al área de proceso.
obtenerObjetivos ()	public: *void*	Obtiene todos los objetivos asociados al área de proceso

Objetivo

public «entity» Class: Representa un objetivo específico o genérico del modelo CMMI-SW.

Objetivo Methods

Method	Type	Notes
calcularValoracion ()	public: *void*	Calcula la valoración asociada al objetivo.
obtenerValoracion ()	public: *void*	Obtiene la valoración asignada al objetivo.
actualizarValoracion ()	public: *void*	Actualiza la valoración del objetivo.
obtenerPractica ()	public: *void*	Obtiene una práctica determinada de las asociadas al objetivo.
obtenerPrácticas ()	public: *void*	Obtiene todas las prácticas asociadas al objetivo

Practica

public «entity» Class: Representa una práctica específica o genérica del modelo CMMI-SW.

Practica Methods

Method	Type	Notes
calcularValoracion ()	public: *void*	Calcula la valoración asociada a la práctica.
actualizarValoracion ()	public: *void*	Actualiza la valoración de la práctica.
obtenerValoracion ()	public: *void*	Obtiene la valoración asociada a la práctica

Observacion

public «entity» Class: Representa una observación relevada durante la evaluación. Las observaciones representan fortalezas y/o debilidades encontradas en la organización.

Observacion Attributes

Attribute	Type	Notes
numero	private :	Numero identificatorio de la observación
tipo	private :	Tipo de observación. Indica si la observación representa una Fortaleza o una Debilidad.
impacto	private :	Indica si la observación tiene un impacto significativo o no dentro de la evaluación.
descripcion	private :	Descripción textual de la observación.

Observacion Methods

Method	Type	Notes
crear ()	public: *void*	Crea una observación.
modificar ()	public: *void*	Modifica los datos de una observación.
eliminar ()	public: *void*	Elimina la observación.

CatalogoGuias

public «entity» Class: Representa el catálogo de guías online para la valoración de los distintos componentes del modelo.

CatalogoGuias Methods

Method	Type	Notes
obtenerGuia ()	public: *void*	Obtiene una guía del catálogo.

6.10 Análisis de la realización de los casos de uso

El siguiente reporte de *Enterprise Architect* muestra las realizaciones de casos de uso de análisis obtenidas como resultado de las actividades "Análisis de los casos de uso" y "Análisis de clases" contempladas en la metodología.

Reporte: Análisis de la realización de los casos de uso

El análisis de la realización de los casos de uso muestra la forma en que interactúan las clases de análisis dentro del sistema para llevar a cabo la funcionalidad descripta en los casos de uso.

Para cada caso de uso del sistema, se efectúa una "realización" de análisis donde se documentan las clases que participan en el mismo y la manera en que interactúan entre sí para llevar a cabo la funcionalidad especificada. Las interacciones entre clases se documentan mediante "Diagramas de colaboración".

El diagrama de la figura 6.21 muestra gráficamente las relaciones de trazabilidad existentes entre las realizaciones de análisis y los casos de uso.

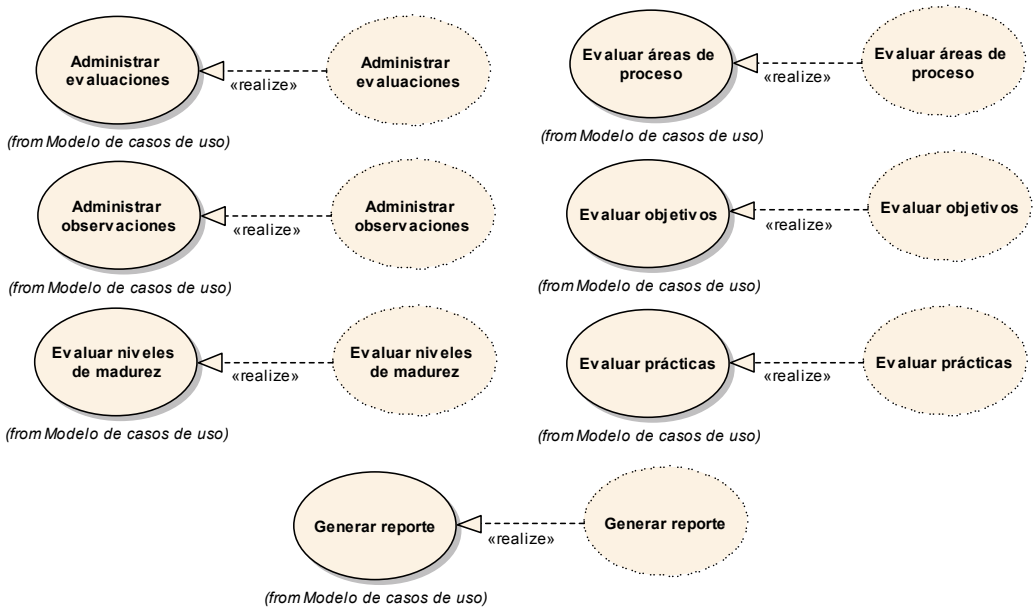

Figura 6.21. Trazabilidad entre realizaciones y casos de uso.

A continuación se detallan una a una todas las realizaciones de análisis.

Administrar evaluaciones

El diagrama de la figura 6.22 muestra las interacciones que se producen en el inicio de una evaluación.

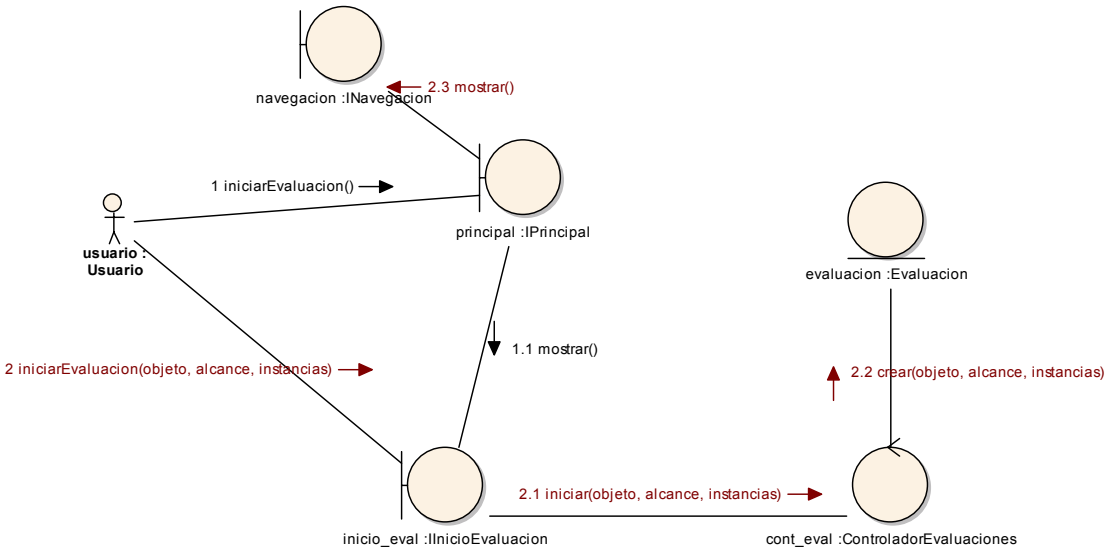

Figura 6.22. Flujo básico: Iniciar evaluación. La secuencia de mensajes 1 cubre la activación de la ventana de inicio de evaluación. La secuencia de mensajes 2 cubre el inicio de la evaluación que se produce cuando el usuario indica los datos y da la orden de iniciar.

Flujo básico: Iniciar evaluación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	iniciarEvaluacion()	usuario	principal	El usuario indica en la interfaz Principal que desea iniciar una nueva evaluación.
1.1	mostrar()	principal	inicio_eval	La interfaz Principal abre la interfaz de Inicio de evaluación
2	iniciarEvaluacion()	usuario	inicio_eval	El usuario suministra los datos (objeto, alcance e instancias) a la interfaz de Inicio de evaluación, y le da la orden de iniciar la evaluación.
2.1	iniciar()	inicio_eval	cont_eval	La interfaz de Inicio de evaluación delega el inicio al controlador.
2.2	crear()	cont_eval	evaluacion	El controlador crea un objeto Evaluacion.
2.3	mostrar()	principal	navegacion	La interfaz Principal abre la interfaz de Navegación del modelo.

El diagrama de la figura 6.23 muestra las interacciones que se producen en el almacenamiento de una evaluación.

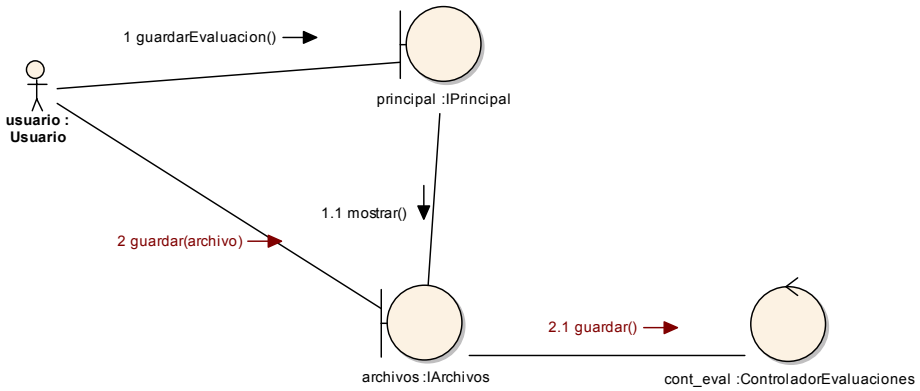

Figura 6.23. Flujo alternativo: Almacenar evaluación. La secuencia de mensajes 1 cubre la activación de la ventana de almacenamiento de evaluación. La secuencia de mensajes 2 cubre el almacenamiento de la evaluación que se produce cuando el usuario indica un nombre de archivo y da la orden de almacenar.

Flujo alternativo: Almacenar evaluación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	guardarEvaluacion()	usuario	principal	El usuario indica en la interfaz Principal que desea guardar la evaluación en curso.
1.1	mostrar()	principal	archivos	La interfaz Principal abre la interfaz de Abrir/Guardar archivos.
2	guardar()	usuario	archivos	El usuario suministra los datos (nombre de archivo) a la interfaz de Archivos, y le da la orden de guardar la evaluación.
2.1	guardar()	archivos	cont_eval	La interfaz de Archivos utiliza el controlador para guardar la evaluación en el archivo indicado.

El diagrama de la figura 6.24 muestra las interacciones que se producen en la recuperación de una evaluación.

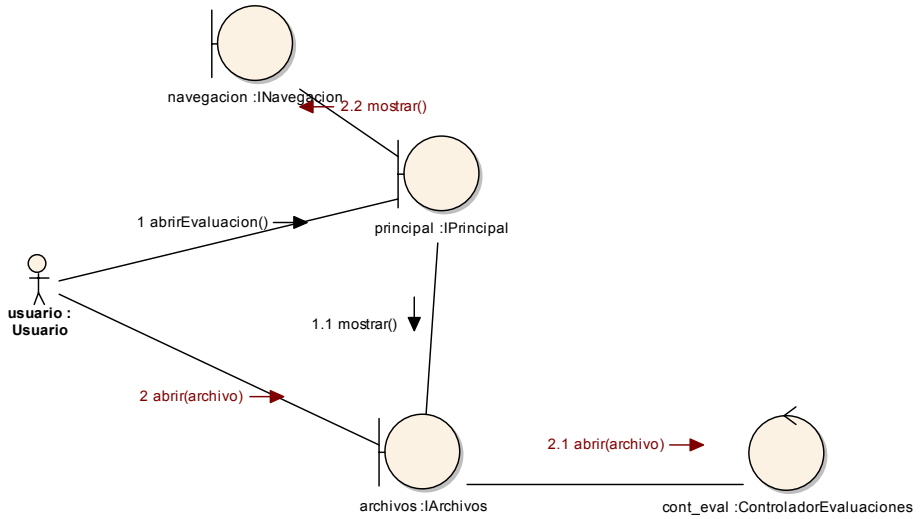

Figura 6.24. Flujo alternativo: Recuperar evaluación. La secuencia de mensajes 1 cubre la activación de la ventana de recuperación de evaluación. La secuencia de mensajes 2 cubre la recuperación de la evaluación que se produce cuando el usuario indica un nombre de archivo y da la orden de recuperar.

Flujo alternativo: Recuperar evaluación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	abrirEvaluacion ()	usuario	principal	El usuario indica en la interfaz Principal que desea abrir una evaluación guardada en archivo.
1.1	mostrar()	principal	archivos	La interfaz Principal abre la interfaz de Abrir/Guardar archivos.
2	abrir()	usuario	archivos	El usuario suministra los datos (nombre de archivo) a la interfaz de Archivos, y le da la orden de abrir la evaluación guardada.
2.1	abrir()	archivos	cont_eval	La interfaz de Archivos utiliza el controlador para abrir la evaluación guardada en el archivo indicado.
2.2	mostrar()	principal	navegacion	La interfaz Principal muestra la interfaz de Navegación del modelo.

El diagrama de la figura 6.25 muestra las interacciones que se producen en el flujo alternativo al inicio y a la recuperación.

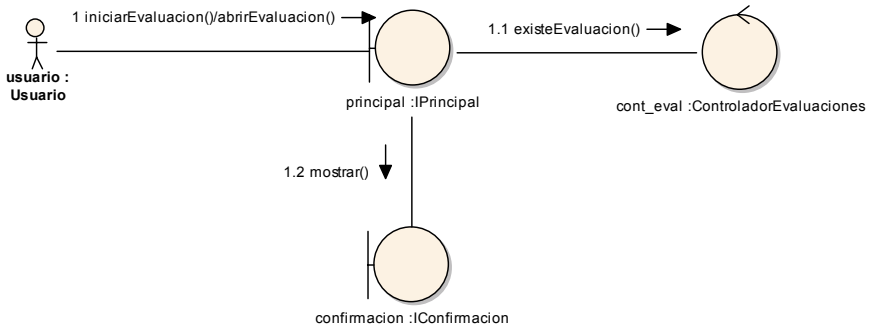

Figura 6.25. Alternativa al Inicio y a la Recuperación.

Alternativa al Inicio y a la Recuperación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	iniciarEvaluacion()/abrirEvaluacion()	usuario	principal	El usuario indica en la interfaz Principal que desea iniciar una nueva evaluación o abrir una evaluación guardada en archivo.
1.1	existeEvaluacion()	principal	cont_eval	La interfaz Principal verifica si existe una evaluación en curso preguntándoselo al controlador.
1.2	mostrar()	principal	confirmacion	En caso de existir una evaluación en curso, la interfaz Principal abre la interfaz de Confirmación para que el usuario indque si desea guardarla.

Administrar observaciones

El diagrama de la figura 6.26 muestra las interacciones que se producen en la creación de una observación.

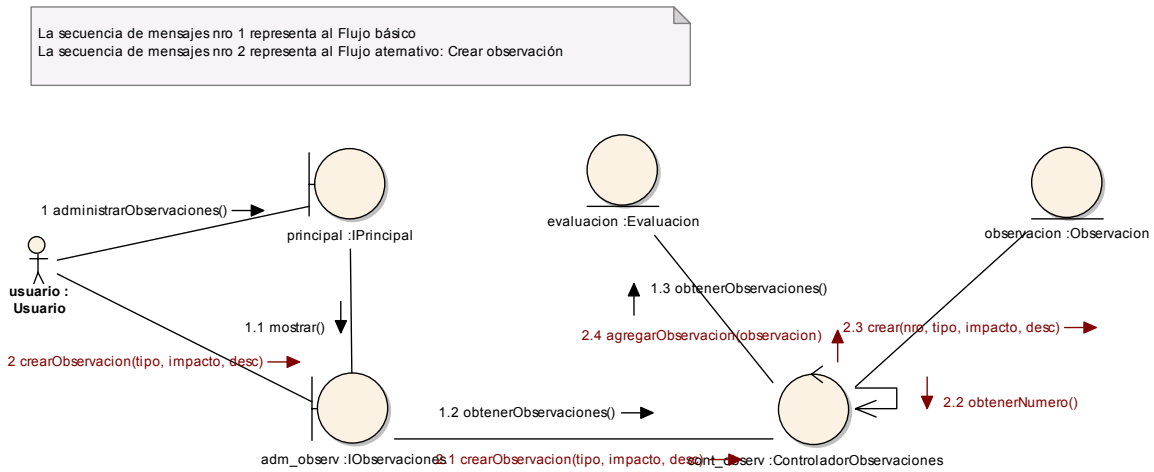

Figura 6.26. Flujo básico + Crear observación. La secuencia de mensajes 1 cubre la activación de la ventana de administración de observaciones. La secuencia de mensajes 2 cubre la creación de una nueva observación, que se produce cuando el usuario indica los datos de entrada y da la orden de crear.

Flujo básico + Crear observación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	administrarObservaciones()	usuario	principal	El usuario indica en la interfaz Principal que desea administrar las observaciones.
1.1	mostrar()	principal	adm_observ	La interfaz Principal muestra la interfaz de Administración de observaciones.
1.2	obtenerObservaciones()	adm_observ	cont_observ	La interfaz de Administración de observaciones solicita la lista de observaciones al controlador de observaciones.
1.3	obtenerObservaciones()	cont_observ	evaluacion	El controlador solicita la lista de observaciones a la evaluación.
2	crearObservacion()	usuario	adm_observ	El usuario ingresa los datos (tipo, impacto y descripción) en la interfaz de administración y da la orden de crear una nueva observación.
2.1	crearObservacion()	adm_observ	cont_observ	La interfaz solicita la creación de una nueva observación al controlador de observaciones.
2.2	obtenerNumero()	cont_observ	cont_observ	El controlador obtiene un número para la observación a crear.
2.3	crear()	cont_observ	observacion	El controlador crea una nueva observación.
2.4	agregarObservacion()	cont_observ	evaluacion	El controlador agrega la observación creada a la evaluación en curso.

El diagrama de la figura 6.27 muestra las interacciones que se producen en la modificación y la eliminación de una observación.

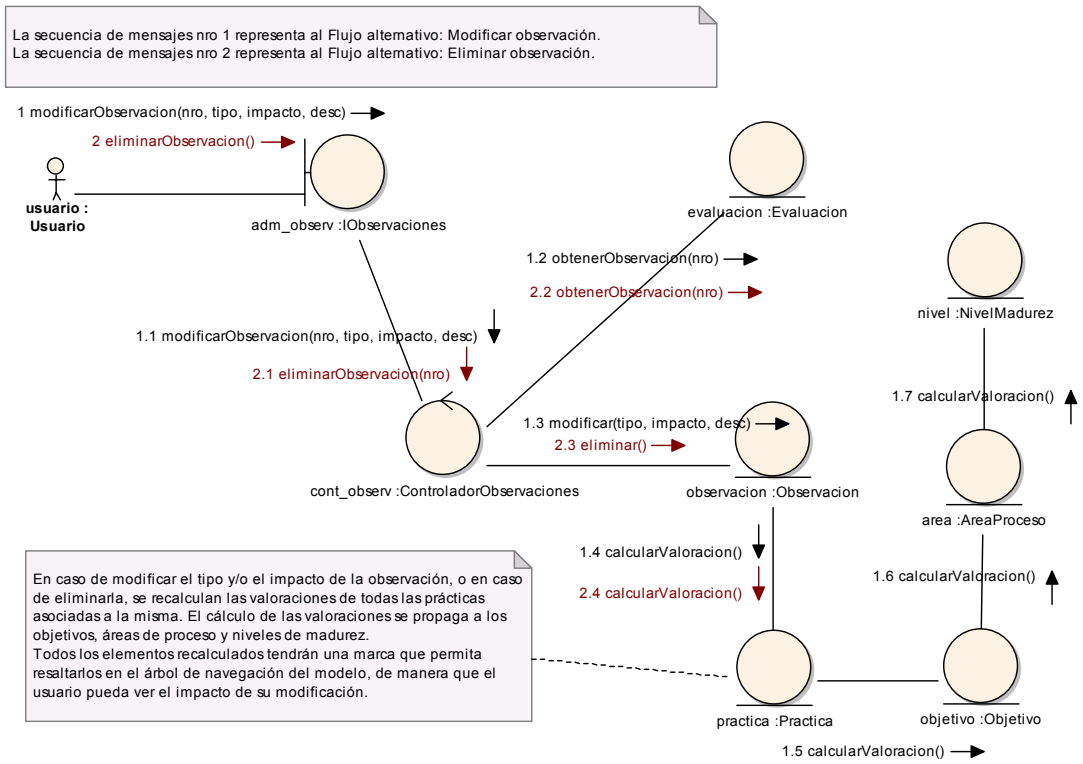

Figura 6.27. Flujos alternativos: Modificar/Eliminar observación. La secuencia de mensajes 1 cubre la modificación de una observación. La secuencia de mensajes 2 cubre la eliminación de una observación.

Flujos alternativos: Modificar/Eliminar observación Collaboration Messages

ID	Message	From Object	To Object	Notes
1	modificarObservacion()	usuario	adm_observ	El usuario selecciona una observación, modifica sus datos y da la orden de modificarla a la interfaz.
1.1	modificarObservacion()	adm_observ	cont_observ	La interfaz solicita al controlador de observaciones la modificación de la observación indicada por el usuario.
1.2	obtenerObservacion()	cont_observ	evaluacion	El controlador solicita la observación a la evaluación.
1.3	modificar()	cont_observ	observacion	El controlador modifica la observación.
1.4	calcularValoracion()	observacion	practica	La observación recalcula las prácticas asociadas a ella.
1.5	calcularValoracion()	practica	objetivo	Las prácticas dan la orden de recalcularse a los objetivos asociados a ellas.
1.6	calcularValoracion()	objetivo	area	Los objetivos dan la orden de recalcularse a las áreas de proceso asociadas a ellos.
1.7	calcularValoracion()	area	nivel	Las áreas de proceso dan la orden de recalcularse a los niveles de madurez que las contienen.
2	eliminarObservacion()	usuario	adm_observ	El usuario selecciona una observación y da la orden de eliminarla a la interfaz.
2.1	eliminarObservacion()	adm_observ	cont_observ	La interfaz solicita al controlador de observaciones la eliminación de la observación indicada por el usuario.

ID	Message	From Object	To Object	Notes
2.2	obtenerObservacion()	cont_observ	evaluacion	El controlador solicita la observación a la evaluación.
2.3	eliminar()	cont_observ	observacion	El controlador elimina la observación.
2.4	calcularValoracion()	observacion	practica	La observación da la orden de recalcularse a las prácticas asociadas a ella.

Evaluar niveles de madurez

El diagrama de la figura 6.28 muestra las interacciones que se producen en la evaluación de un nivel de madurez.

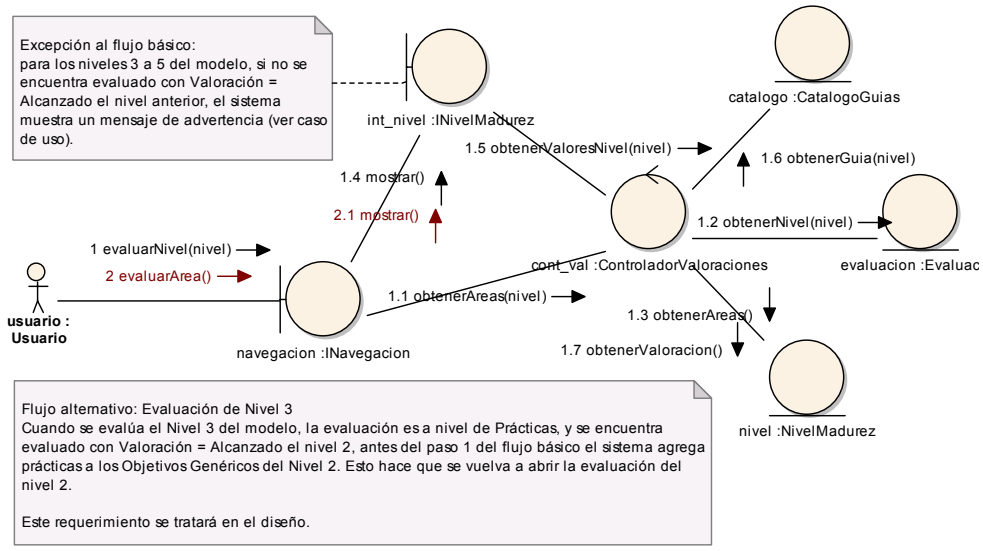

Figura 6.28. Flujo básico + alternativos. La secuencia de mensajes 1 cubre la inicialización del nivel de madurez con las áreas de proceso. La secuencia de mensajes 2 cubre la activación de la ventana de evaluación.

Flujo básico + alternativos Collaboration Messages

ID	Message	From Object	To Object	Notes
1	evaluarNivel()	usuario	navegacion	El usuario selecciona un nivel de madurez en el árbol y e indica que desea evaluarlo a la interfaz de Navegación del modelo.
1.1	obtenerAreas()	navegacion	cont_val	La interfaz solicita al controlador de valoraciones las áreas de proceso correspondientes al nivel seleccionado.
1.2	obtenerNivel()	cont_val	evaluacion	El controlador de valoraciones solicita el nivel a la evaluación.
1.3	obtenerAreas()	cont_val	nivel	El controlador de valoraciones solicita al nivel sus áreas de proceso.
1.4	mostrar()	navegacion	int_nivel	Luego de desplegar las áreas, la interfaz de Navegación muestra la interfaz de Evaluación de niveles de madurez.
1.5	obtenerValoresNivel()	int_nivel	cont_val	La interfaz de evaluación solicita las guías y valoraciones al controlador de valoraciones.

ID	Message	From Object	To Object	Notes
1.6	obtenerGuia()	cont_val	catalogo	El controlador de valoraciones solicita al catálogo la guía de valoración correspondiente al nivel de madurez.
1.7	obtenerValoracion()	cont_val	nivel	El controlador obtiene la valoración asociada al nivel de madurez.
2	evaluarArea()	usuario	navegacion	El usuario evalúa una a una las áreas de proceso contenidas en el nivel de madurez.
2.1	mostrar()	navegacion	int_nivel	Una vez que el usuario termina de evaluar todas las áreas de proceso, al posicionarse sobre el nivel de madurez en la interfaz de navegación, la misma muestra la interfaz de Evaluación de niveles, donde podrá verse la valoración concluida.

Evaluar áreas de proceso

El diagrama de la figura 6.29 muestra las interacciones que se producen en la evaluación de un área de proceso.

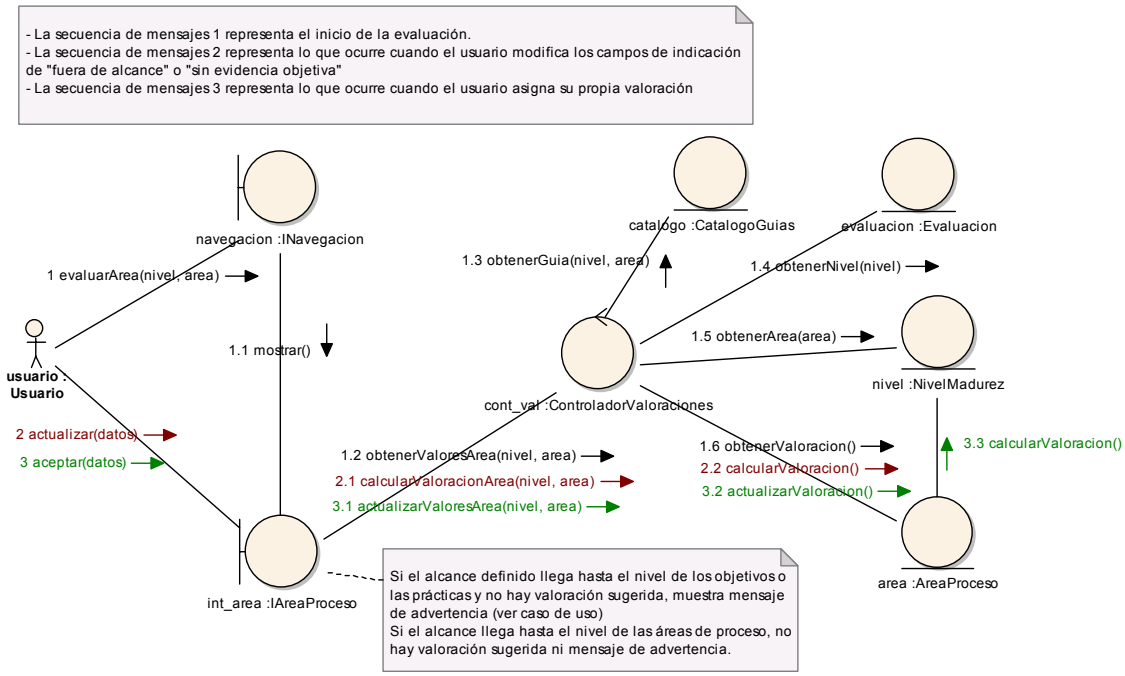

Figura 6.29. Flujo básico + alternativos. La secuencia de mensajes 1 cubre la activación de la ventana de evaluación. La secuencia de mensajes 2 cubre la actualización de la valoración sugerida, que se produce cuando el usuario modifica los datos de entrada. La secuencia de mensajes 3 cubre la asignación de la valoración por parte del usuario.

Flujo básico + alternativos Collaboration Messages

ID	Message	From Object	To Object	Notes
1	evaluarArea()	usuario	navegacion	El usuario selecciona un área de proceso en el árbol e indica que desea evaluarla a la interfaz de Navegación del modelo.
1.1	mostrar()	navegacion	int_area	La interfaz de Navegación muestra la interfaz de Evaluación de áreas de proceso.
1.2	obtenerValoresArea()	int_area	cont_val	La interfaz de evaluación solicita las guías y valoraciones al controlador de valoraciones.
1.3	obtenerGuia()	cont_val	catalogo	El controlador de valoraciones solicita al catálogo la guía de valoración correspondiente al área de proceso.
1.4	obtenerNivel()	cont_val	evaluacion	El controlador de valoraciones solicita el nivel a la evaluación.
1.5	obtenerArea()	cont_val	nivel	El controlador de valoraciones solicita el área de proceso al nivel.
1.6	obtenerValoracion()	cont_val	area	El controlador obtiene la valoración asociada al área de proceso.
2	actualizar()	usuario	int_area	El usuario modifica los datos de la interfaz ("fuera de alcance" o "sin evidencia objetiva"), lo que genera una orden de actualización de la valoración sugerida.
2.1	calcularValoracionArea()	int_area	cont_val	La interfaz de evaluación solicita al controlador de valoracioens que se recalcule la valoración sugerida.
2.2	calcularValoracion()	cont_val	area	El controlador solicita al área que recalcule su valoración sugerida.
3	aceptar()	usuario	int_area	El usuario asigna una valoración con su justificación y da la orden de aceptar a la interfaz de evaluación.
3.1	actualizarValoresArea()	int_area	cont_val	La interfaz solicita al controlador de valoraciones que actualice la valoración del área.
3.2	actualizarValoracion()	cont_val	area	El controlador actualiza la valoración del área.
3.3	calcularValoracion()	area	nivel	En caso de que la nueva valoración asignada al área sea distinta de la previa, el área informa a su nivel que debe recalcular su valoración concluía.

Evaluar objetivos

El diagrama de la figura 6.30 muestra las interacciones que se producen en la evaluación de un objetivo.

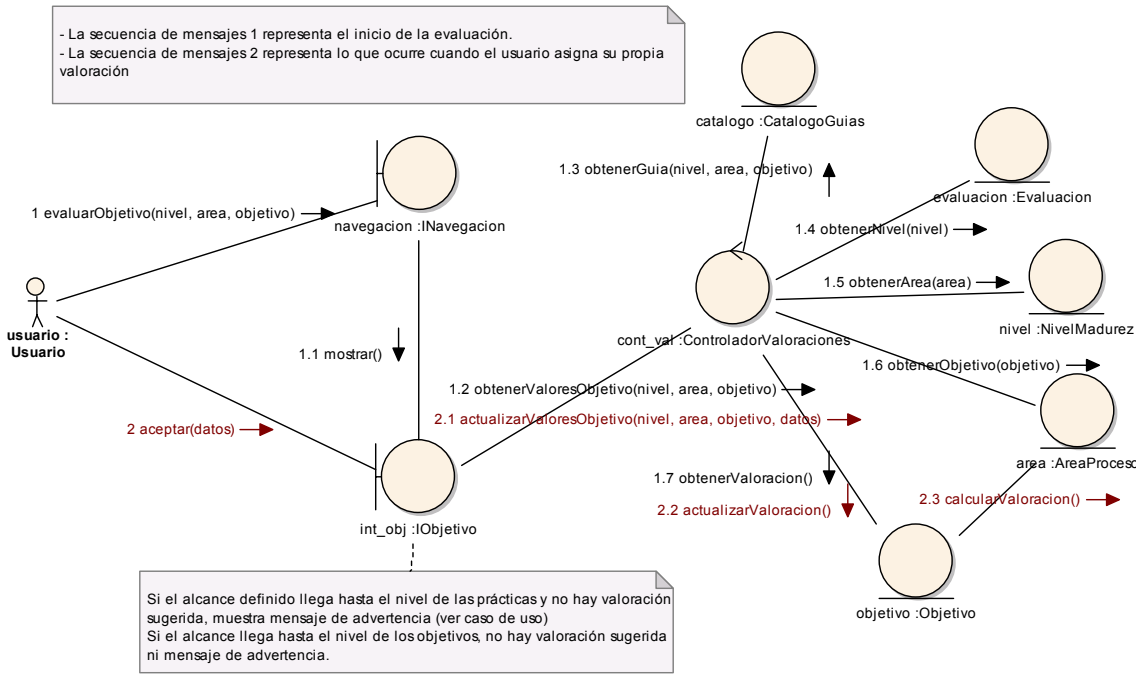

Figura 6.30. Flujo básico + alternativos. La secuencia de mensajes 1 cubre la activación de la ventana de evaluación. La secuencia de mensajes 2 cubre la asignación de la valoración por parte del usuario.

Flujo básico + alternativos Collaboration Messages

ID	Message	From Object	To Object	Notes
1	evaluarObjetivo()	usuario	navegacion	El usuario selecciona un objetivo en el árbol e indica que desea evaluarlo a la interfaz de Navegación del modelo.
1.1	mostrar()	navegacion	int_obj	La interfaz de Navegación muestra la interfaz de Evaluación de objetivos.
1.2	obtenerValoresObjetivo()	int_obj	cont_val	La interfaz de evaluación solicita las guías y valoraciones al controlador de valoraciones.
1.3	obtenerGuia()	cont_val	catalogo	El controlador de valoraciones solicita al catálogo la guía de valoración correspondiente al objetivo.
1.4	obtenerNivel()	cont_val	evaluacion	El controlador de valoraciones solicita el nivel a la evaluación.
1.5	obtenerArea()	cont_val	nivel	El controlador de valoraciones solicita el área de proceso al nivel.
1.6	obtenerObjetivo()	cont_val	area	El controlador de valoraciones solicita el objetivo al área de proceso.
1.7	obtenerValoracion()	cont_val	objetivo	El controlador obtiene la valoración asociada al objetivo.
2	aceptar()	usuario	int_obj	El usuario asigna una valoración con su justificación y da la orden de aceptar a la interfaz de evaluación.
2.1	actualizarValoresObjetivo()	int_obj	cont_val	La interfaz solicita al controlador de valoraciones que actualice la valoración del objetivo.
2.2	actualizarValoracion()	cont_val	objetivo	El controlador actualiza la valoración del objetivo.
2.3	calcularValoracion()	objetivo	area	En caso de que la nueva valoración asignada al objetivo sea distinta de la previa, el objetivo informa a su área que debe recalcular su valoración sugerida. El área a su vez notificará a su nivel.

Evaluar prácticas

El diagrama de la figura 6.31 muestra las interacciones que se producen en la evaluación de una práctica.

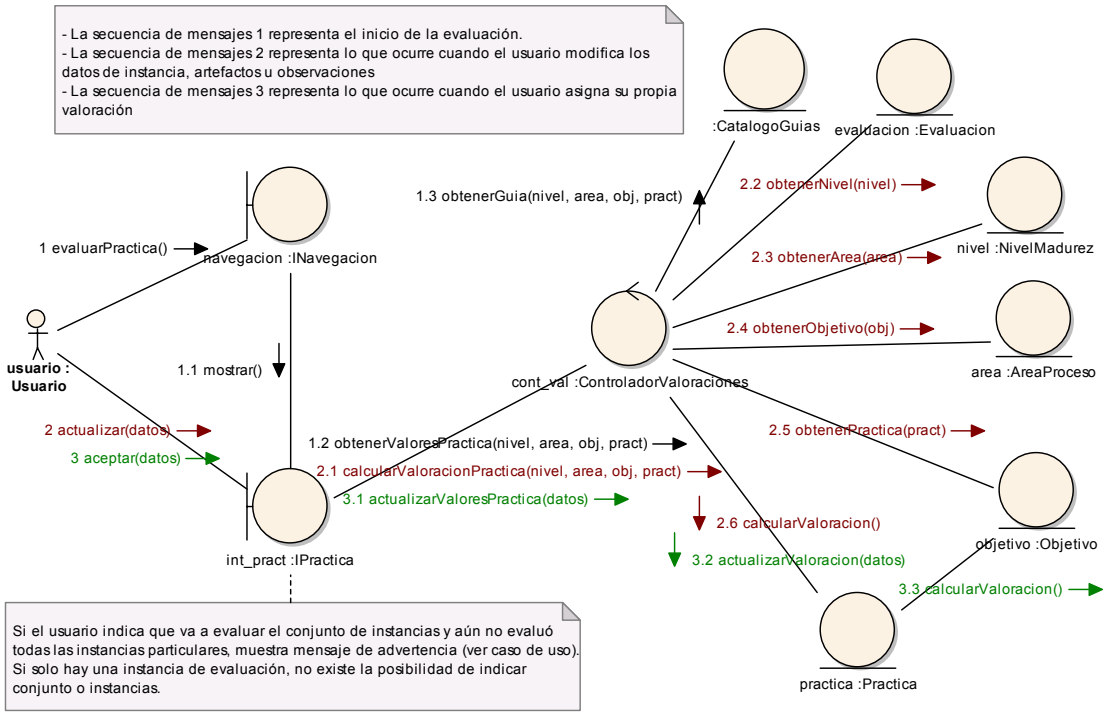

Figura 6.31. Flujo básico + alternativos. La secuencia de mensajes 1 cubre la activación de la ventana de evaluación. La secuencia de mensajes 2 cubre la actualización de la valoración sugerida, que se produce cuando el usuario modifica los datos de entrada. La secuencia de mensajes 3 cubre la asignación de la valoración por parte del usuario.

Flujo básico + alternativos Collaboration Messages

ID	Message	From Object	To Object	Notes
1	evaluarPractica()	usuario	navegacion	El usuario selecciona una práctica en el árbol e indica que desea evaluarla a la interfaz de Navegación del modelo.
1.1	mostrar()	navegacion	int_pract	La interfaz de Navegación muestra la interfaz de Evaluación de prácticas.
1.2	obtenerValoresPractica()	int_pract	cont_val	La interfaz de evaluación solicita las guías y valoraciones al controlador de valoraciones.
1.3	obtenerGuia()	cont_val		El controlador de valoraciones solicita al catálogo la guía de valoración correspondiente a la práctica.
2	actualizar()	usuario	int_pract	El usuario modifica los datos de la interfaz (datos de instancia, de artefactos o de observaciones), lo que genera una orden de actualización de la valoración sugerida.
2.1	calcularValoracionPractica()	int_pract	cont_val	La interfaz de evaluación solicita al controlador de valoraciones que se recalcule la valoración sugerida.

ID	Message	From Object	To Object	Notes
2.2	obtenerNivel()	cont_val	evaluacion	El controlador solicita el nivel a la evaluación.
2.3	obtenerArea()	cont_val	nivel	El controlador solicita el área al nivel.
2.4	obtenerObjetivo()	cont_val	area	El controlador solicita el objetivo al área.
2.5	obtenerPractica()	cont_val	objetivo	El controlador solicita la práctica al objetivo.
2.6	calcularValoracion()	cont_val	practica	El controlador solicita a la práctica que recalcule su valoración sugerida.
3	aceptar()	usuario	int_pract	El usuario asigna una valoración con su justificación y da la orden de aceptar a la interfaz de evaluación.
3.1	actualizarValoresPractica()	int_pract	cont_val	La interfaz solicita al controlador de valoraciones que actualice la valoración de la práctica.
3.2	actualizarValoracion()	cont_val	practica	El controlador actualiza la valoración de la práctica.
3.3	calcularValoracion()	practica	objetivo	En caso de que la nueva valoración asignada a la práctica sea distinta de la previa, la práctica informa a su objetivo que debe recalcular su valoración concluía. A su vez, el objetivo informa a su área y el área a su nivel.

Generar reporte

El diagrama de la figura 6.32 muestra las interacciones que se producen en la generación de un reporte.

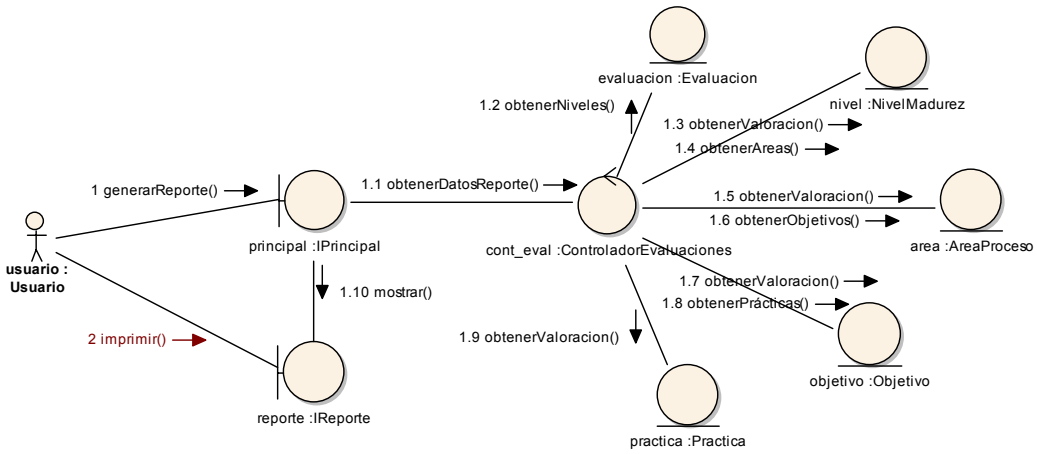

Figura 6.32. Flujo básico. La secuencia de mensajes 1 cubre la generación del reporte. La secuencia de mensajes 2 cubre la impresión del mismo por parte del usuario.

Flujo básico Collaboration Messages

ID	Message	From Object	To Object	Notes
1	generarReporte()	usuario	principal	El usuario indica en la interfaz Principal que desea generar un reporte de la evaluación en curso
1.1	obtenerDatosReporte()	principal	cont_eval	La interfaz Principal solicita los datos para llenar el reporte al Controlador de evaluaciones
1.2	obtenerNiveles()	cont_eval	evaluacion	El controlador solicita los niveles de madurez a la evaluación
1.3	obtenerValoracion()	cont_eval	nivel	El controlador obtiene las valoraciones de cada nivel de madurez
1.4	obtenerAreas()	cont_eval	nivel	El controlador obtiene las áreas de cada nivel
1.5	obtenerValoracion()	cont_eval	area	El controlador obtiene las valoraciones de cada área de proceso
1.6	obtenerObjetivos()	cont_eval	area	El controlador obtiene los objetivos asociados a cada área de proceso
1.7	obtenerValoracion()	cont_eval	objetivo	El controlador obtiene las valoraciones de cada objetivo
1.8	obtenerPrácticas()	cont_eval	objetivo	El controlador obtiene las prácticas asociadas al objetivo
1.9	obtenerValoracion()	cont_eval	practica	El controlador obtiene las valoraciones de cada práctica
1.10	mostrar()	principal	reporte	La interfaz Principal muestra la interfaz de Reportes
2	imprimir()	usuario	reporte	El usuario imprime el reporte

6.11 Análisis de consistencia

6.11.1 Verificación de los modelos

Se verificó la calidad de los distintos modelos por separado, a fines de garantizar que eran adecuados de acuerdo a la técnica seguida para su elaboración. Como resultado de la verificación, se efectuaron correcciones en el Modelo del negocio, el Modelo de clases de análisis, y en las Realizaciones de análisis.

6.11.2 Resultado del análisis de consistencia

Se comprobó la coherencia entre los distintos modelos de acuerdo a las trazabilidades presentadas en el apartado 6.1 (Modelos y trazabilidad).

La comprobación se llevó a cabo mediante un conjunto de matrices de trazabilidad, generadas desde la herramienta *Enterprise Architect*.

Catálogo de requisitos vs Modelo del negocio

La matriz de la tabla 6.2 muestra en las filas los requerimientos del Catálogo de requisitos y en las columnas las actividades del Modelo del negocio. Como puede verse en la misma, cada una de las actividades tiene su correspondencia con algún requisito. Existen además otros requisitos que no tienen correspondencia con actividades del negocio. Estos requisitos fueron agregados como consecuencia de la interacción con el usuario y cubren aspectos de funcionamiento operativo (RF-005, RF-006 y RF-007) o bien aspectos no funcionales (RN-001 al RN-004).

	1. Evaluar prácticas	2. Evaluar objetivos	3. Evaluar áreas de proceso	4. Evaluar nivel de madurez
RF-001: Evaluación de prácticas	X			
RF-002: Evaluación de objetivos		X		
RF-003: Evaluación de áreas de proceso			X	
RF-004: Evaluación del nivel de madurez				X
RF-005: Generación de reporte				
RF-006: Almacenamiento de evaluación en archivos				
RF-007: Recuperación de archivos de evaluación.				
RN-001: Interfaz gráfica				
RN-002: Guías online				
RN-003: Asistencia en la evaluación				
RN-004: Soporte de sistemas operativos				

Tabla 6.2. Catálogo de requisitos vs. Modelo del negocio.

Modelo de casos de uso vs Catálogo de requisitos

La matriz de la tabla 6.3 muestra en las filas los casos de uso del Modelo de casos de uso y en las columnas los requisitos del Catálogo de requisitos. Como puede verse en la misma, cada uno de los requisitos tiene su correspondencia con algún caso de uso del sistema.

	RF-001: Evaluación de prácticas	RF-002: Evaluación de objetivos	RF-003: Evaluación de áreas de proceso	RF-004: Evaluación del nivel de madurez	RF-005: Generación de reporte	RF-006: Almacenamiento de evaluación en archivos	RF-007: Recuperación de archivos de evaluación.	RN-001: Interfaz gráfica	RN-002: Guías online	RN-003: Asistencia en la evaluación	RN-004: Soporte de sistemas operativos
Administrar evaluaciones						X	X	X			X
Administrar observaciones	X							X			X
Evaluar niveles de madurez				X				X	X	X	X
Evaluar objetivos		X						X	X	X	X
Evaluar prácticas	X							X	X	X	X
Evaluar áreas de proceso			X					X	X	X	X
Generar reporte					X			X			X

Tabla 6.3. Modelo de casos de uso vs. Catálogo de requisitos.

Prototipos de interfaz vs Modelo de casos de uso

La matriz de la tabla 6.4 muestra en las filas las pantallas del modelo Prototipos de interfaz y en las columnas los casos de uso del Modelo de casos de uso. Como puede verse en la misma, cada una de las pantallas tiene su correspondencia con algún caso de uso del sistema.

	Administrar evaluaciones	Administrar observaciones	Evaluar niveles de madurez	Evaluar objetivos	Evaluar prácticas	Evaluar áreas de proceso	Generar reporte
Abrir/Guardar evaluación	X						
Administración de Observaciones		X					
Evaluación de Nivel de madurez			X				
Evaluación de Objetivo				X			
Evaluación de Práctica					X		
Evaluación de Área de proceso						X	
Inicio de evaluación	X						
Navegación del modelo			X	X	X	X	
Principal	X	X					X
Reporte							X

Tabla 6.4. Prototipos de interfaz vs. Modelo de casos de uso.

Prototipos de interfaz vs Modelo de navegación de interfaz

La matriz de la tabla 6.5 muestra en las filas las pantallas del modelo Prototipos de interfaz y en las columnas las pantallas del Modelo de navegación de interfaz. Como puede verse en la misma, las pantallas de ambos modelos son coherentes entre sí.

	Abrir evaluación	Administración de observaciones	Evaluación de Nivel de madurez	Evaluación de Objetivo	Evaluación de Práctica	Evaluación de Área de proceso	Guardar evaluación	Inicio de evaluación	Navegación del modelo	Principal	Reporte
Abrir/Guardar evaluación	X						X				
Administración de Observaciones		X									
Evaluación de Nivel de madurez			X								
Evaluación de Objetivo				X							
Evaluación de Práctica					X						
Evaluación de Área de proceso						X					
Inicio de evaluación								X			
Navegación del modelo									X		
Principal										X	
Reporte											X

Tabla 6.5. Prototipos de interfaz vs. Modelo de de navegación de interfaz.

Realizaciones de análisis vs Modelo de casos de uso

La matriz de la tabla 6.6 muestra en las filas las colaboraciones del modelo Realizaciones de análisis y en las columnas los casos de uso del Modelo de casos de uso. Como puede verse en la misma, cada uno de los casos de uso ha sido realizado en una colaboración.

	Administrar evaluaciones	Administrar observaciones	Evaluar niveles de madurez	Evaluar objetivos	Evaluar prácticas	Evaluar áreas de proceso	Generar reporte
Administrar evaluaciones	X						
Administrar observaciones		X					
Evaluar niveles de madurez			X				
Evaluar objetivos				X			
Evaluar prácticas					X		
Evaluar áreas de proceso						X	
Generar reporte							X

Tabla 6.6. Realizaciones de análisis vs. Modelo de casos de uso.

Modelo de clases de análisis vs Realizaciones de análisis

La matriz de la tabla 6.7 muestra en las filas las colaboraciones del modelo Realizaciones de análisis y en las columnas las clases del Modelo de clases de análisis. Como puede verse en la misma, cada clase ha participado en al menos una colaboración.

	AreaProceso	CatalogoGuias	ControladorEvaluaciones	ControladorObservaciones	ControladorValoraciones	Evaluacion	IArchivos	IAreaProceso	IConfirmacion	IInicioEvaluacion	INavegacion	INivelMadurez	IObjetivo	IObservaciones	IPractica	IPrincipal	IReporte	NivelMadurez	Objetivo	Observacion	Practica
Administrar evaluaciones			X			X	X		X	X	X					X					
Administrar observaciones	X			X		X								X		X		X	X	X	X
Evaluar niveles de madurez		X			X	X					X	X						X			
Evaluar objetivos	X	X			X	X					X		X					X	X		
Evaluar prácticas	X	X			X	X					X				X			X	X		X
Evaluar áreas de proceso	X	X			X	X		X			X							X			
Generar reporte	X		X			X										X	X	X	X		X

Tabla 6.7. Modelo de clases de análisis vs. Realizaciones de análisis.

Además de estas comprobaciones, se verificó que cada una de las operaciones de las clases de análisis corresponda a un mensaje de los diagramas de colaboración. No se grafica esa matriz debido a que su extensión es excesiva.

6.11.3 Validación con el usuario

Se validaron con el usuario los siguientes modelos:

- Catálogo de requisitos
- Modelo de casos de uso
- Prototipos de interfaz

El usuario estuvo de acuerdo con los requerimientos planteados para el sistema y la manera en que los mismos se cubren mediante la funcionalidad expresada en los casos de uso. Asimismo, estuvo de acuerdo en la interacción planeada para el sistema a través de los prototipos de interfaz.

7. DISEÑO DEL SISTEMA

En este capítulo se documentan los productos de salida del proceso DSI (Diseño del Sistema de Información) de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

7.1 Modelos y trazabilidad

El proceso DSI (Diseño del Sistema de Información) cubre un conjunto de modelos relacionados entre sí. Este documento muestra cada uno de esos modelos mediante reportes extraídos de la herramienta *Enterprise Architect* [EA, 2004].

El diagrama de la figura 7.1 muestra las relaciones de trazabilidad que existen entre los distintos modelos presentados en este documento, y con los modelos resultantes del proceso de Análisis del Sistema de Información.

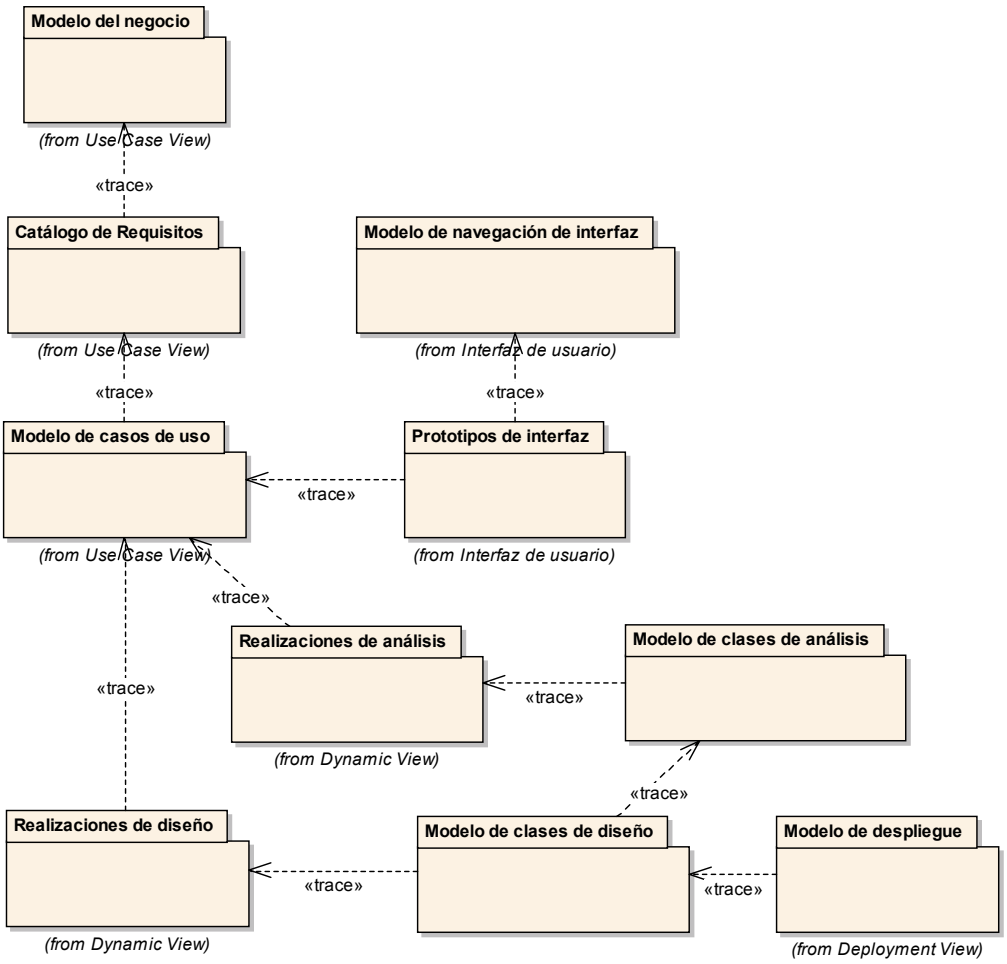

Figura 7.1: trazabilidad entre los distintos modelos. La parte inferior del diagrama muestra los modelos generados durante el proceso DSI (Realizaciones de diseño, Modelo de clases de diseño, Modelo de despliegue).

7.2 Diseño de la arquitectura del sistema

7.2.1 Particionamiento físico del sistema de información

El particionamiento físico del sistema se documenta mediante un Modelo de despliegue UML [Booch & Jacobson & Rumbaugh, 1998]. A continuación se muestra un reporte generado en *Enterprise Architect* mostrando los contenidos de dicho modelo.

Reporte: Modelo de despliegue

Este modelo muestra la arquitectura del sistema desde el punto de vista físico. Para ello, el modelo contempla los siguientes elementos:

- Nodos de procesamiento

- Dispositivos de hardware
- Comunicaciones entre nodos y con dispositivos
- Componentes de software empaquetados en unidades instalables

El diagrama de la figura 7.2 muestra los elementos involucrados en el sistema.

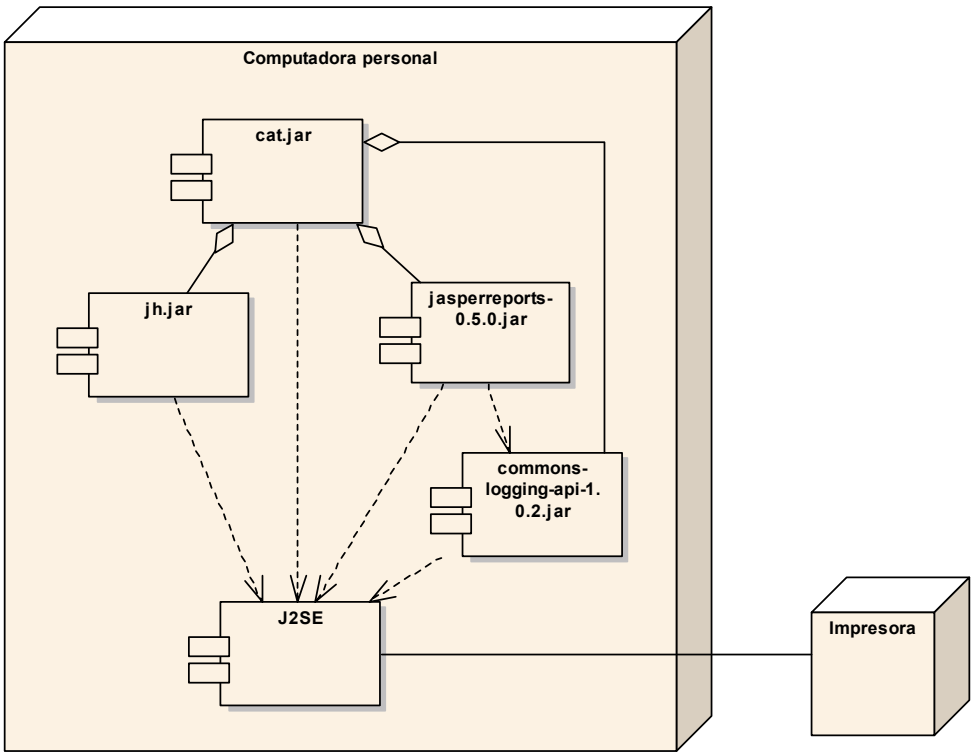

Figura 7.2. Diagrama de despliegue.

A continuación, se describen los elementos del diagrama.

cat.jar

public Component: Este componente es el sistema completo. Se trata de una aplicación Java del tipo standalone con interfaz gráfica de usuario del tipo Swing [J2SE, 2004].

jasperreports-0.5.0.jar

public Component: Este componente es el Runtime de JasperReports [JasperReports, 2004]. Se utiliza para la generación e impresión de reportes.

commons-logging-api-1.0.2.jar

public Component: Este componente es la librería de vuelco de mensajes de log utilizada por JasperReports.

jh.jar

public Component: Este componente es el Runtime de JavaHelp [JavaHelp, 2004]. Se utiliza para todas las ayudas online y contextuales de la aplicación.

J2SE

public Component: Este componente es el Runtime (entorno de ejecución) de la plataforma Java. Es necesario como base para la ejecución del sistema, ya que brinda la máquina virtual y las librerías utilizadas por la aplicación.

Computadora personal

public Node: Es el único nodo de procesamiento necesario para la ejecución del sistema completo (CAT). Características:

- Procesador: mínimo Pentium III o equivalente
- Memoria: mínimo 64 MBytes
- Espacio en disco: 3 MBytes

Impresora

public Node: Para imprimir los reportes generados por la aplicación, la instalación deberá disponer de una impresora. La comunicación con la impresora se realiza por intermedio de la plataforma Java (J2SE).

7.2.2 Descripción de subsistemas de diseño

Los subsistemas de diseño se documentan como parte del Modelo de diseño UML dentro de lo que es la Arquitectura lógica del sistema. A continuación se muestra un reporte generado con *Enterprise Architect* en el cual se muestra la arquitectura mencionada.

Reporte: Arquitectura lógica

El diagrama de la figura 7.3 muestra la arquitectura del sistema desde el punto de vista lógico, contemplando los distintos paquetes y subsistemas que lo componen. Los mismos se encuentran distribuídos en dos capas: una capa correspondiente a los subsistemas y paquetes específicos del negocio (superior) y otra correspondiente a los subsistemas y paquetes de soporte (inferior).

La arquitectura tomada como base para el funcionamiento de la aplicación está dada por el patrón de diseño *Model View Controller* [Hunt, J. 2001]. En el paquete *mvc* se describe el funcionamiento de la misma.

En el diagrama se destacan con la leyenda API (Interfaz de Programación de Aplicación, en inglés *Application Programming Interface*) aquellos paquetes de terceras partes utilizados en el sistema (en particular para el soporte de Reportes y Ayudas).

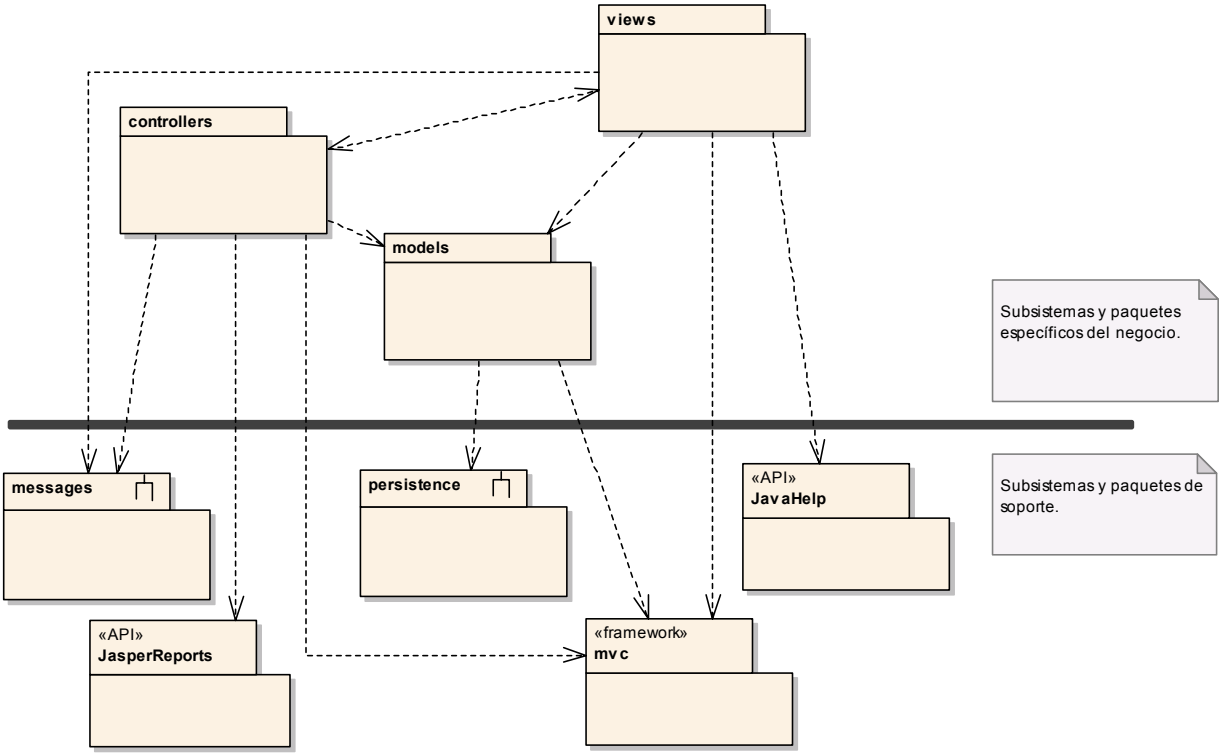

Figura 7.3. Arquitectura lógica del sistema.

A continuación se describe cada uno de los componentes del diagrama.

mvc

Este paquete contiene un conjunto de clases e interfaces que definen un framework para implementar el patrón de diseño MVC (*Model View Controller*) [Hunt, J. 2001]. Las clases de *views*, *controllers* y *model* extienden a las clases e implementan las interfaces definidas en el framework.

El patrón de diseño MVC permite separar por un lado la interfaz gráfica de usuario, por otro el control de las acciones que el mismo realiza en el sistema, y por otro el modelo de información que hace a la lógica de negocios.

Las vistas (*views*) y los controladores (*controllers*) se conocen mutuamente y conocen a su vez al modelo (*model*), pero el modelo no conoce ni a las vistas ni a los controladores. Esto significa que pueden utilizarse diferentes interfaces de usuario sobre un mismo modelo, y que además esas interfaces pueden cambiar sin impacto sobre el mismo.

views

Este paquete contiene todas las clases del tipo *view* (vista) correspondientes al patrón de diseño MVC. Estas clases brindan las interfaces gráficas de usuario y son derivadas de las clases del paquete Swing (J2SE).

controllers

Este paquete contiene todas las clases del tipo *controller* (controlador) correspondientes al patrón de diseño MVC. Estas clases procesan las acciones efectuadas por el usuario en las views y las traducen a eventos o invocaciones en el model (lógica de negocios de la aplicación).

models

Este paquete contiene todas las clases del tipo *model* (modelo) correspondientes al patrón de diseño MVC. Estas clases brindan toda la funcionalidad específica del negocio, y constituyen en sí mismas la lógica de negocios de la aplicación.

persistence

Este subsistema contiene las clases encargadas de la persistencia de los objetos del negocio (*model*) en el sistema de archivos. El subsistema expone una interfaz para el almacenamiento y la recuperación de objetos contra el sistema de archivos.

messages

Este subsistema contiene todas las clases encargadas del manejo de los mensajes que se muestran al usuario. El subsistema expone una interfaz mediante la cual se pueden recuperar mensajes de texto a partir de códigos.

JasperReports

Este subsistema representa al producto JasperReports utilizado para la generación de reportes. El mismo es una API que brinda un conjunto de clases utilizables directamente en la aplicación.

JavaHelp

Este subsistema representa al producto JavaHelp utilizado para el manejo de ayudas dentro de la aplicación. El mismo es una API que brinda un conjunto de clases utilizables directamente en la aplicación.

7.3 Entorno tecnológico del sistema

Los elementos de infraestructura técnica que dan soporte al sistema de información son los siguientes:

- Elementos de Hardware

  - Computadora personal Pentium II o superior
  - 64 Mbytes de memoria RAM mínimo
  - Espacio mínimo en disco de 3 Mbytes
- Elementos de software
  - Plataforma J2SE 1.4.2 o superior
  - Cualquier sistema operativo que soporte la plataforma J2SE

El sistema no necesita de ninguna infraestructura de comunicaciones, ya que se trata de una herramienta del tipo *standalone*.

7.4 Diseño de subsistemas y paquetes de soporte

En esta sección se documentan los diferentes subsistemas y paquetes de soporte que conforman el sistema completo.

Cada subsistema o paquete se documenta contemplando dos puntos de vista básicos: su estructura y su funcionamiento. La estructura se documenta mediante diagramas de clase y el funcionamiento mediante diagramas de secuencia.

7.4.1 mvc – Framework Model View Controller

La arquitectura de la aplicación está basada en el patrón de diseño *Model View Controller* (MVC). El paquete *mvc* ubicado en la capa de los subsistemas y paquetes de soporte, contiene un framework que implementa este patrón de diseño. Los paquetes de la capa específica del negocio (capa superior) contienen clases que extienden la funcionalidad de este framework de manera específica al negocio.

A continuación se muestra un reporte extraído de la herramienta *Enterprise Architect* donde se explica la estructura y el comportamiento del framework MVC.

Reporte: mvc - Estructura

Este paquete contiene un conjunto de clases e interfaces que definen un framework para implementar el patrón de diseño MVC (*Model View Controller*). Las clases de *views*, *controllers* y *model* extienden a las clases e implementan las interfaces definidas en el framework.

El patrón de diseño *Model View Controller* permite separar por un lado la interfaz gráfica de usuario, por otro el control de las acciones que el mismo realiza en el sistema, y por otro el modelo de información que hace a la lógica de negocios.

Las vistas (*views*) y los controladores (*controllers*) se conocen mutuamente y conocen a su vez al modelo (*model*), pero el modelo no conoce ni a las vistas ni a los controladores. Esto significa que pueden utilizarse diferentes interfaces de usuario sobre

un mismo modelo, y que además esas interfaces pueden cambiar sin impacto sobre el mismo.

El diagrama de la figura 7.4 muestra las clases e interfaces que conforman el framework MVC utilizado como arquitectura base de la aplicación.

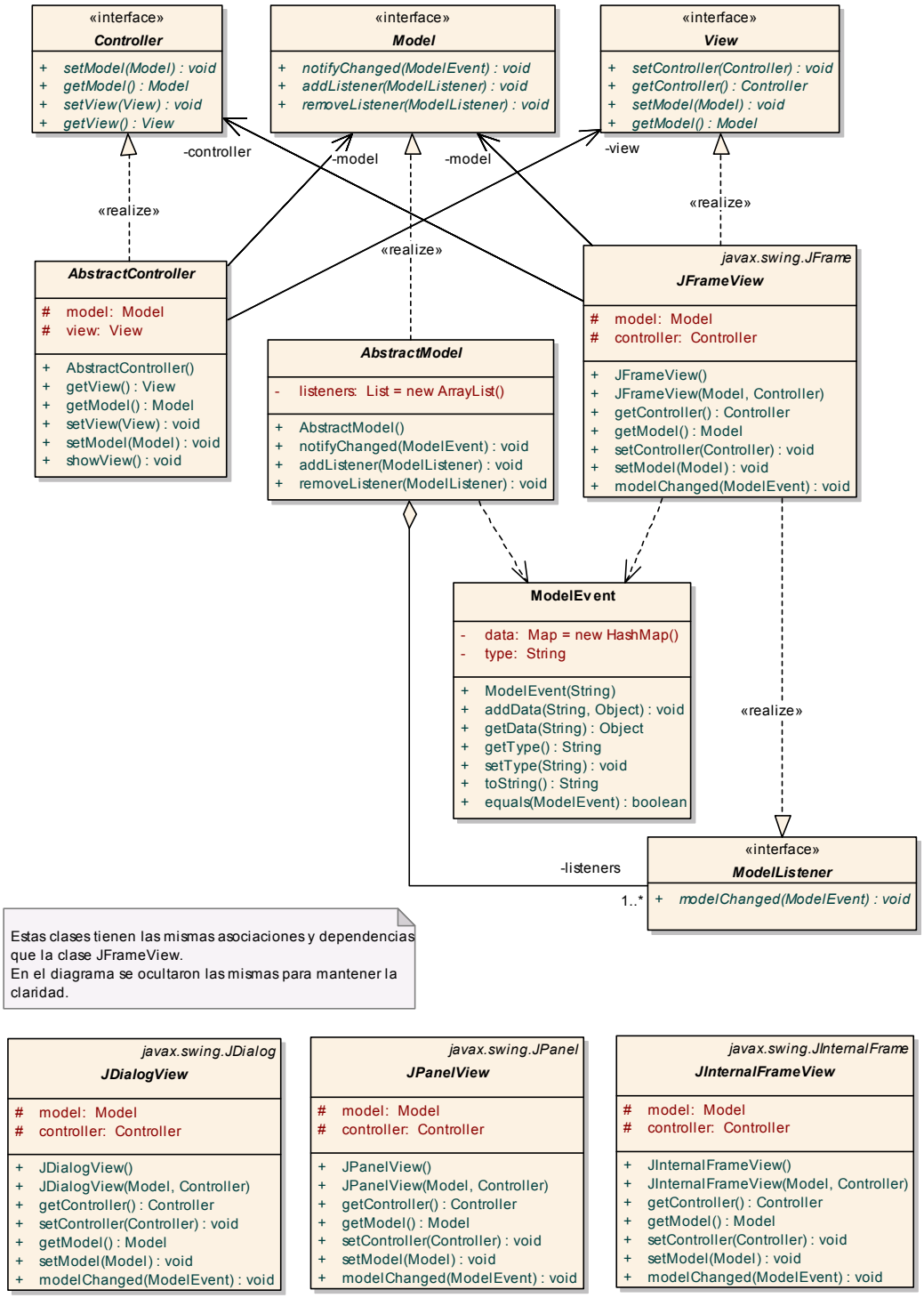

Figura 7.4. Clases del framework MVC.

A continuación se describe cada una de las clases presentadas en el diagrama.

AbstractController

public abstract Class

Implements: Controller. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *Controller*. Las implementaciones de default permiten asociar un modelo y una vista al controlador, y obtener posteriormente el modelo y la vista asociados.

AbstractController Attributes

Attribute	Type	Notes
model	protected : Model	Modelo asociado al controlador.
view	protected : View	Vista asociada al controlador.

AbstractController Methods

Method	Type	Notes
AbstractController ()	public:	Constructor de default
getView ()	«property get» public: View	Obtiene la vista asociada al controlador
getModel ()	«property get» public: Model	Obtiene el modelo asociado al controlador
setView (View)	«property set» public: void	param: newVal [View - in] Vista a setear Setea la vista asociada el controlador
setModel (Model)	«property set» public: void	param: newVal [Model - in] Modelo a setear Setea el modelo asociado al controlador
showView ()	public: void	Muestra la vista asociada

AbstractModel

public abstract Class

Implements: Model. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *Model*. Provee métodos para registrar y eliminar suscriptores interesados en los cambios del modelo, y una implementación de default para la notificación de los cambios. La notificación de default consiste en recorrer la lista de suscriptores y enviar a cada uno de ellos el mensaje *modelChanged* con un *ModelEvent* indicando los cambios.

AbstractModel Attributes

Attribute	Type	Notes
listeners	private : List	Lista de objetos ModelListener interesados en los cambios del modelo. Initial Value: new ArrayList();

AbstractModel Methods

Method	Type	Notes
AbstractModel ()	public:	Constructor de default
notifyChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula información sobre los cambios efectuados en el modelo. Notifica a los objetos interesados acerca de los cambios en el modelo.
addListener (ModelListener)	public: void	param: listener [ModelListener - in] Objeto a agregar en la lista. Agrega un objeto a la lista de suscriptores del modelo. Al agregar el objeto, le notifica con un ModelEvent de tipo "Welcome".
removeListener (ModelListener)	public: void	param: listener [ModelListener - in] Objeto a eliminar de la lista. Elimina un objeto de la lista de suscriptores del modelo. Al eliminar el objeto, le notifica con un ModelEvent de tipo BYE.

JFrameView

public abstract Class

Implements: ModelListener, View, javax.swing.JFrame. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *View*. Las implementaciones de default permiten asociar un modelo y un controlador a la vista, y obtener posteriormente el modelo y el controlador asociados. Esta clase hereda de javax.swing.JFrame, pero pueden definirse otras que extiendan a otros componentes del paquete Swing (JPane, JDialog, etc.) Implementa la interfaz *ModelListener* para suscribirse a los cambios en el modelo.

JFrameView Attributes

Attribute	Type	Notes
model	protected : Model	Modelo asociado a la vista.
controller	protected : Controller	Controlador asociado a la vista.

JFrameView Methods

Method	Type	Notes
JFrameView ()	public:	Constructor de default
JFrameView (Model, Controller)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociados.
getController ()	«property get» public: Controller	Permite obtener el controlador asociado
getModel ()	«property get» public: Model	Permite obtener el modelo asociado
setController (Controller)	«property set» public: void	param: newVal [Controller - in] Controlador a asociar Permite asociar un controlador
setModel (Model)	«property set» public: void	param: newVal [Model - in] Modelo a asociar Permite asociar un modelo
modelChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula la información de los cambios producidos en el modelo. Notificación de cambios en el modelo.

JDialogView

public abstract Class

Implements: ModelListener, View, javax.swing.JDialog. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *View*. Las implementaciones de default permiten asociar un modelo y un controlador a la vista, y obtener posteriormente el modelo y el controlador asociados. Esta clase hereda de javax.swing.JDialog Implementa la interfaz *ModelListener* para suscribirse a los cambios en el modelo.

JDialogView Attributes

Attribute	Type	Notes
model	protected : Model	Modelo asociado a la vista.
controller	protected : Controller	Controlador asociado a la vista.

JDialogView Methods

Method	Type	Notes
JDialogView ()	public:	Constructor de default
JDialogView (Model, Controller)	public:	param: model [Model - in] param: controller [Controller - in] Constructor que recibe el modelo y el controlador asociados
getController ()	«property get» public: Controller	Permite obtener el controlador asociado
setController (Controller)	«property set» public: void	param: newVal [Controller - in] Controlador a asociar. Permite asociar un controlador
getModel ()	«property get» public: Model	Permite obtener el modelo asociado
setModel (Model)	«property set» public: void	param: newVal [Model - in] Modelo a asociar. Permite asociar un modelo
modelChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula la información de los cambios producidos en el modelo. Notificación de cambios en el modelo.

JPanelView

public abstract Class

Implements: ModelListener, View, javax.swing.JPanel. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *View*. Las implementaciones de default permiten asociar un modelo y un controlador a la vista, y obtener posteriormente el modelo y el controlador asociados. Esta clase hereda de javax.swing.JPanel. Implementa la interfaz *ModelListener* para suscribirse a los cambios en el modelo.

JPanelView Attributes

Attribute	Type	Notes
model	protected : Model	Modelo asociado a la vista.
controller	protected : Controller	Controlador asociado a la vista.

JPanelView Methods

Method	Type	Notes
JPanelView ()	public:	Constructor de default
JPanelView (Model, Controller)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociados.
getController ()	«property get» public: Controller	Permite obtener el controlador asociado
getModel ()	«property get» public: Model	Permite obtener el modelo asociado
setController (Controller)	«property set» public: void	param: newVal [Controller - in] Controlador a asociar Permite asociar un controlador
setModel (Model)	«property set» public: void	param: newVal [Model - in] Modelo a asociar Permite asociar un modelo
modelChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula la información de los cambios producidos en el modelo. Notificación de cambios en el modelo.

JInternalFrameView

public abstract Class

Implements: ModelListener, View, javax.swing.JInternalFrame. : Clase abstracta que brinda implementaciones de default para los comportamientos definidos en la interfaz *View*. Las implementaciones de default permiten asociar un modelo y un controlador a la vista, y obtener posteriormente el modelo y el controlador asociados. Esta clase hereda de javax.swing. JInternalFrame. Implementa la interfaz *ModelListener* para suscribirse a los cambios en el modelo.

JInternalFrameView Attributes

Attribute	Type	Notes
model	protected : Model	Modelo asociado a la vista.
controller	protected : Controller	Controlador asociado a la vista.

JInternalFrameView Methods

Method	Type	Notes
JInternalFrameView ()	public:	Constructor de default
JInternalFrameView (Model, Controller)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociados.
getController ()	«property get» public: Controller	Permite obtener el controlador asociado
getModel ()	«property get» public: Model	Permite obtener el modelo asociado
setController (Controller)	«property set» public: void	param: newVal [Controller - in] Controlador a asociar. Permite asociar un controlador
setModel (Model)	«property set» public: void	param: newVal [Model - in] Modelo a asociar. Permite asociar un modelo
modelChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula la información de los cambios producidos en el modelo. Notificación de cambios en el modelo.

ModelEvent

public Class: Representa los eventos de notificación mediante los cuales los objetos suscriptores se enteran de los cambios en los modelos. Contiene un mapa para alojar todos los datos necesarios, en pares nombre-valor.

ModelEvent Attributes

Attribute	Type	Notes
data	private : Map	Mapa con pares nombre-valor donde se alojan todos los cambios efectuados en un modelo. Initial Value: new HashMap();
type	private : String	Tipo de evento

ModelEvent Methods

Method	Type	Notes
ModelEvent (String)	public:	param: type [String - in] Constructor que recibe el tipo de evento
addData (String, Object)	public: void	param: name [String - in] Nombre del dato a agregar. param: value [Object - in] Valor del dato a agregar. Agrega un para nombre-valor al mapa de datos.
getData (String)	public: Object	param: name [String - in] Nombre del dato a obtener. Obtiene el valor asociado a un dato almacenado en el mapa.
getType ()	public: String	Retorna el tipo de evento
setType (String)	public: void	param: string [String - in] Tipo a setear Setea el tipo de evento
toString ()	public: String	Reimplementación del método toString para mostrar los contenidos del evento
equals (ModelEvent)	public: boolean	param: other [ModelEvent - in] Reimplementación del método equals para comparar eventos

Controller

public abstract «interface» Interface: Define el comportamiento común para todos los *controllers* del patrón MVC. Todas las clases del tipo controller (paquete controllers) implementan esta interfaz. El comportamiento común consiste en asociar un modelo y una vista al controlador, y obtener el modelo y la vista asociados.

Controller Interfaces

Method	Type	Notes
setModel (Model)	public: void	param: model [Model - in] Modelo a asociar con el controlador Permite asociar un modelo a este controlador.
getModel ()	public: Model	Permite obtener el modelo asociado a este controlador.
setView (View)	public: void	param: view [View - in] Vista a asociar con el controlador. Permite asociar una vista a este controlador.
getView ()	public: View	Permite obtener la vista asociada al controlador.

Model

public abstract «interface» Interface: Define el comportamiento común para todos los *models* del patrón MVC. Todas las clases del tipo model (paquete models) implementan esta interfaz. El comportamiento común consiste en brindar un método para notificar a los objetos interesados acerca de los cambios en el modelo.

Model Interfaces

Method	Type	Notes
notifyChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula información sobre los cambios efectuados en el modelo. Notifica a los objetos interesados acerca de los cambios en el modelo.
addListener (ModelListener)	public: void	param: listener [ModelListener - in] Objeto a agregar en la lista. Agrega un objeto a la lista de suscriptores del modelo.
removeListener (ModelListener)	public: void	param: listener [ModelListener - in] Objeto a eliminar de la lista. Elimina un objeto de la lista de suscriptores del modelo.

View

public abstract «interface» Interface: Define el comportamiento común para todas las *views* del patrón MVC. Todas las clases del tipo view (paquete views) implementan esta interfaz. El comportamiento común consiste en asociar un modelo y un controlador a la vista, y obtener el modelo y el controlador asociados.

View Interfaces

Method	Type	Notes
setController (Controller)	public: void	param: controller [Controller - in] Controlador a asociar a la vista. Permite asociar un controlador a esta vista.
getController ()	public: Controller	Permite obtener el controlador asociado a la vista.
setModel (Model)	public: void	param: model [Model - in] Modelo a asociar con la vista. Permite asociar un modelo a esta vista.
getModel ()	public: Model	Permite obtener el modelo asociado a esta vista.

ModelListener

public abstract «interface» Interface: Define el comportamiento común que deben implementar todos los objetos que deseen suscribirse como interesados en los

cambios del modelo. El comportamiento común consiste en brindar un método para aceptar las notificaciones de cambio generadas desde el modelo, utilizando como medio de notificación un *ModelEvent*.

ModelListener Interfaces

Method	Type	Notes
modelChanged (ModelEvent)	public: void	param: event [ModelEvent - in] Evento del modelo. Encapsula la información de los cambios producidos en el modelo. Notificación de cambios en el modelo.

Reporte: mvc - Funcionamiento

Aquí se contemplan los aspectos de funcionamiento dinámico del patrón de diseño *Model View Controller*.

El funcionamiento se explica mediante diagramas de interacción donde se muestra la comunicación entre los objetos del framework en tiempo de ejecución.

En el diagrama de la figura 7.5 se muestran las interacciones que se producen durante la inicialización de una tríada de objetos que responden al patrón MVC.

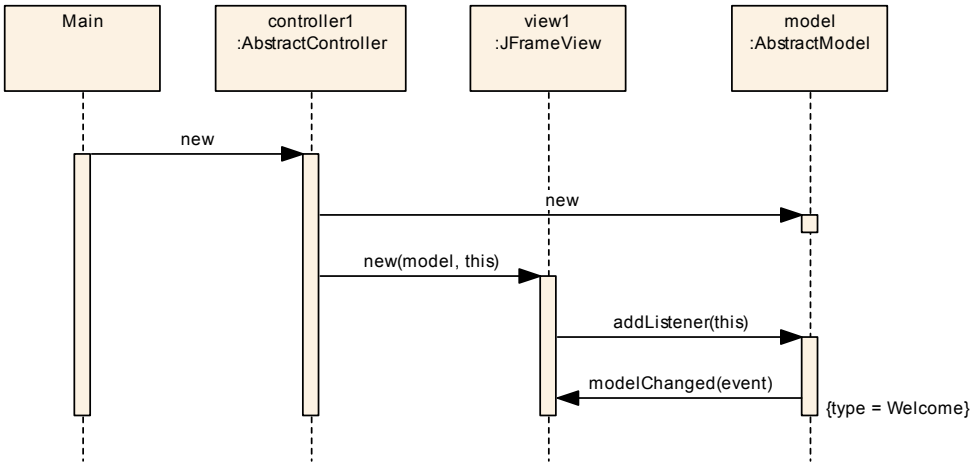

Figura 7.5. Inicialización. El programa principal (o una parte del mismo) crea una instancia de algún controlador específico (en el diagrama representado mediante el controlador abstracto). El controlador crea u obtiene una referencia del modelo específico (representado en el diagrama mediante el modelo abstracto) al cual debe transmitir las acciones del usuario, y una instancia de la vista específica a presentar al usuario (representada en el diagrama mediante una vista abstracta). Al crear la vista, le pasa como parámetro el modelo asociado a la misma y una referencia a sí mismo. La vista asocia el modelo y el controlador a sus atributos, y se registra como observador (*listener*) del modelo recibido como parámetro, ante lo cual el modelo genera una notificación de bienvenida (*ModelEvent* con tipo "Welcome").

En el diagrama de la figura 7.6 se muestran las interacciones que se producen cuando el usuario realiza alguna acción sobre una vista de una tríada de objetos que responden al patrón MVC.

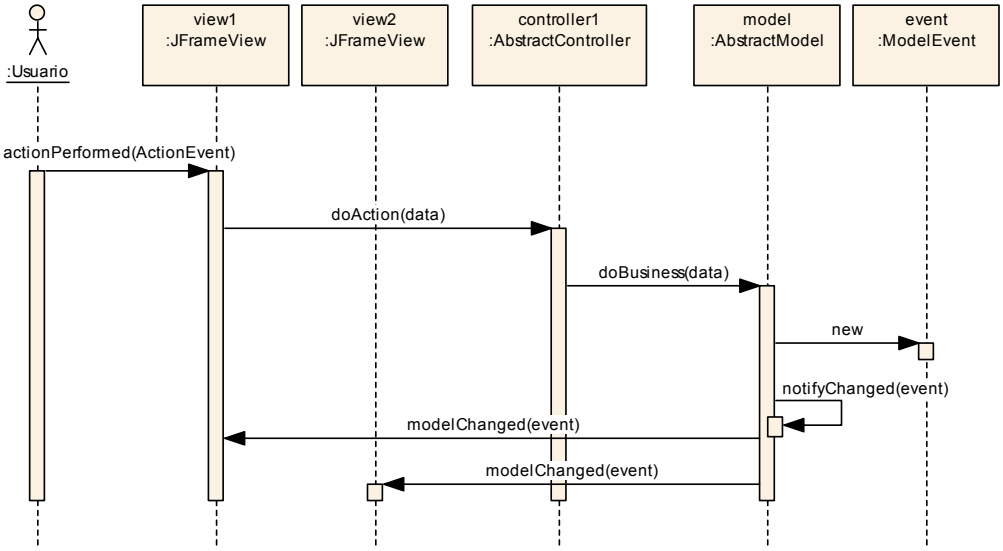

Figura 7.6. Procesamiento de una acción de usuario. La acción del usuario se traduce en una notificación de evento Swing (en el diagrama representado por un *ActionEvent*). La vista que recibe el evento lo delega a su controlador asociado, invocando algún método provisto por el mismo para procesar acciones de usuario (en el diagrama representado por doAction(data)). El controlador traduce la acción del usuario en una invocación a un método de negocios de su modelo asociado (representado en el diagrama como doBusiness(data)). El método de negocios genera cambios en el modelo, ante lo cual el modelo crea un evento de notificación (*ModelEvent*) y notifica a todos los objetos interesados en los cambios. La notificación hace que se refresquen todas las vistas suscriptas al modelo que se acaba de modificar.

7.4.2 persistence – Subsistema de persistencia de archivos

Este subsistema se encarga de la persistencia de la información en archivos. A continuación, se muestra un reporte de *Enterprise Architect* donde se explica su estructura y funcionamiento.

Reporte: persistence - Estructura

Este subsistema contiene las clases encargadas de la persistencia de los objetos del negocio (*model*) en el sistema de archivos.

El subsistema expone una interfaz para el almacenamiento y la recuperación de objetos contra el sistema de archivos.

El diagrama de la figura 7.7 muestra la estructura interna del subsistema.

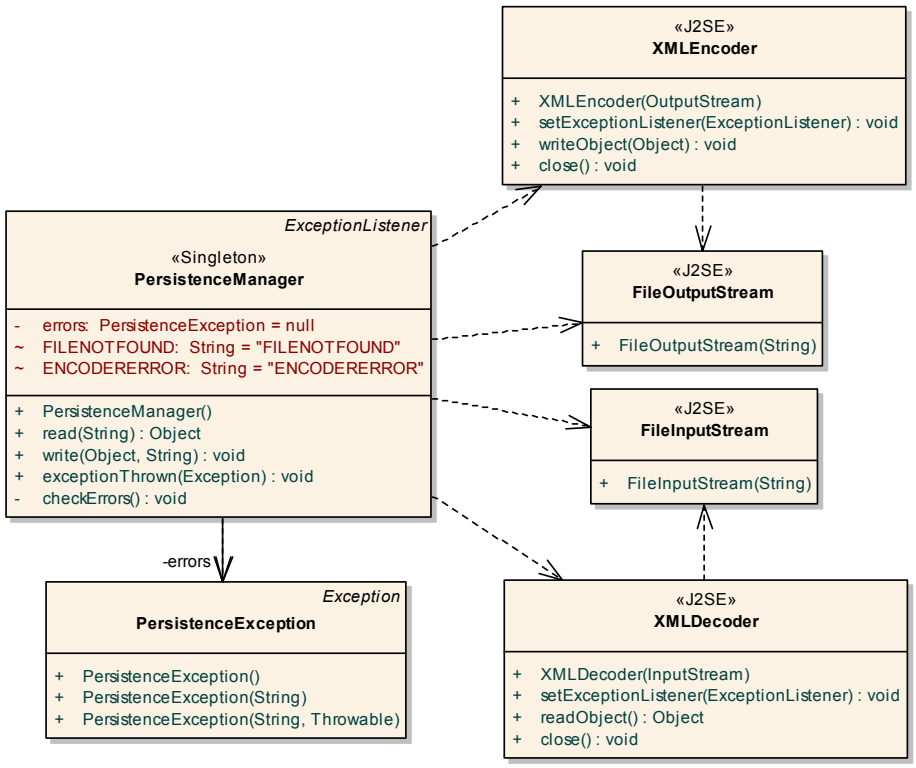

Figura 7.7. Clases del subsistema persistence.

A continuación se describe cada una de las clases que lo conforman.

PersistenceManager

public «Singleton» Class

Implements: ExceptionListener. : Esta clase se encarga de manejar la serialización y deserialización de un grafo de objetos hacia y desde un archivo. Para ello utiliza los servicios de las clases *XMLEncoder* y *XMLDecoder* del paquete java.beans. La única condición que deben cumplir los objetos a serializar es la de seguir las convenciones de los componentes del tipo JavaBean [JavaBeans, 2004]. Esto significa que deben proveer un constructor de default (sin parámetros), y tantos métodos "get" y "set" como atributos posea el objeto. *PersistenceManager* implementa la interfaz *ExceptionListener* para escuchar las excepciones arrojadas por el *XMLEncoder* y el *XMLDecoder*. La clase implementa el patrón de diseño *Singleton*, lo que significa que existe un único objeto de esta clase en toda la aplicación.

PersistenceManager Attributes

Attribute	Type	Notes
errors	private : PersistenceException	Excepción a arrojar en caso de errores Initial Value: null;
FILENOTFOUND	package : String	Código de error en apertura de archivo Initial Value: "FILENOTFOUND";
ENCODERERROR	package : String	Código de error en uso de encoder o decoder Initial Value: "ENCODERERROR";

PersistenceManager Methods

Method	Type	Notes
PersistenceManager ()	public:	Constructor de default
read (String)	public: Object	param: file [String - in] Archivo de entrada. Lee un objeto desde un archivo
write (Object, String)	public: void	param: object [Object - in] Objeto a almacenar. param: file [String - in] Archivo de destino. Escribe un objeto en un archivo.
exceptionThrown (Exception)	public: void	param: exception [Exception - in] Excepción producida Método invocado cada vez que se produce una excepción en un objeto sobre el cual el PersistenceManager está escuchando
checkErrors ()	private: void	Verifica el estado del indicador y arroja una excepción en caso existir errores

PersistenceException

public Class

Implements: Exception. : Excepción arrojada por el subsistema cuando no puede leer o escribir los objetos en los archivos. Extiende a la clase *Exception* de J2SE.

PersistenceException Methods

Method	Type	Notes
PersistenceException (String)	public:	param: message [String - in] Mensaje asociado a la excepción Constructor a partir de un mensaje. Delega al constructor de la superclase.
PersistenceException ()	public:	
PersistenceException (String, Throwable)	public:	param: message [String - in] Mensaje asociado a la excepción param: cause [Throwable - in] Excepción original Constructor a partir de un mensaje y una excepción. Delega al constructor de la superclase.

XMLEncoder

public «J2SE» Class: Clase encargada de convertir objetos en documentos XML. Pertenece a las librerías de la plataforma J2SE.

XMLEncoder Methods

Method	Type	Notes
XMLEncoder (OutputStream)	public:	param: stream [OutputStream - in] Stream de salida Constructor que recibe como parámetro un stream de salida
setExceptionListener (ExceptionListener)	public: void	param: listener [ExceptionListener - in] Objeto interesado en las excepciones Registra un objeto como listener de excepciones
writeObject (Object)	public: void	param: object [Object - in] Objeto a convertir y escribir Convierte el objeto recibido como parámetro a XML y lo escribe en el stream recibido en el constructor
close ()	public: void	Cierra el encoder

FileOutputStream

public «J2SE» Class: Clase que permite escribir información en un archivo. Pertenece a las librerías de la plataforma J2SE.

FileOutputStream Methods

Method	Type	Notes
FileOutputStream (String)	public:	param: file [String - in] Nombre de archivo Constructor a partir de un nombre de archivo

XMLDecoder

public «J2SE» Class: Clase encargada de convertir documentos XML en objetos. Pertenece a las librerías de la plataforma J2SE.

XMLDecoder Methods

Method	Type	Notes
XMLDecoder (InputStream)	public:	param: stream [InputStream - in] Stream de entrada Constructor que recibe como parámetro un stream de entrada
setExceptionListener (ExceptionListener)	public: void	param: listener [ExceptionListener - in] Objeto interesado en las excepciones Registra un objeto como listener de excepciones
readObject ()	public: Object	Lee un documento XML del stream recibido en el constructor y lo convierte en un objeto
close ()	public: void	Cierra el decoder

FileInputStream

public «J2SE» Class: Clase que permite leer información desde un archivo. Pertenece a las librerías de la plataforma J2SE.

FileInputStream Methods

Method	Type	Notes
FileInputStream (String)	public:	param: file [String - in] Nombre de archivo Constructor a partir de un nombre de archivo

Reporte: persistence - Funcionamiento

Aquí se contemplan los aspectos de funcionamiento dinámico del subsistema de persistencia en archivos.

El funcionamiento se explica mediante diagramas de interacción donde se muestra la comunicación entre los objetos del subsistema en tiempo de ejecución.

En el diagrama de la figura 7.8 se muestran las interacciones que se producen cuando una clase cliente solicita al *PersistenceManager* que almacene un objeto.

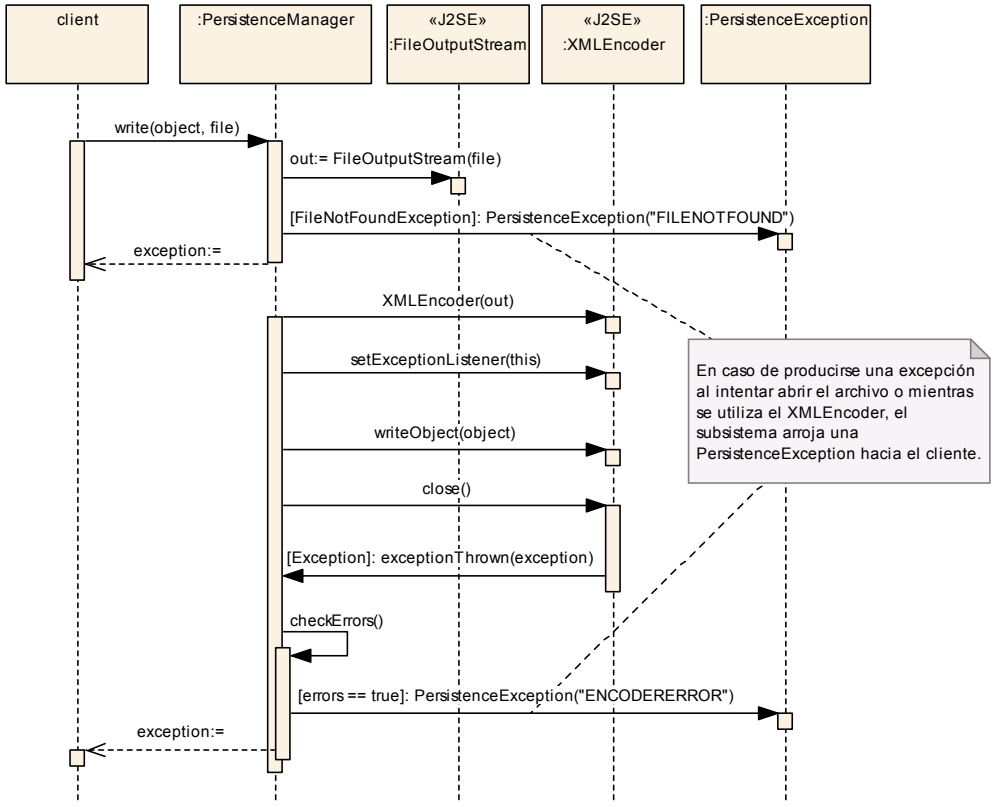

Figura 7.8. Escritura de un objeto en un archivo. La clase cliente envía el mensaje *write* al *PersistenceManager* pasándole como argumentos el objeto a almacenar y el nombre del archivo de destino. El *PersistenceManager* crea un flujo de salida (*FileOutputStream*) utilizando las librerías de java. En caso de producirse una excepción (*FileNotFoundException*), instancia una *PersistenceException* con

código de mensaje FILENOTFOUND y la arroja al cliente.
En caso de no haber excepciones, el *PersistenceManager* crea una instancia de *XMLEncoder* pasándole como parámetro el flujo de salida, y se registra como interesado (*listener*) en las excepciones del mismo. Finalmente, el *PersistenceManager* envía el mensaje *writeObject* al *XMLEncoder*, pasándole el objeto a almacenar, y luego lo cierra mediante el mensaje *close*.
En caso de producirse una excepción en alguno de los métodos del *XMLEncoder*, el mismo invoca al método *exceptionThrown* de su *listener* (el *PersistenceManager*), quien se encarga de setear el indicador de errores (atributo *errors*) en true. Finalmente, el *PersistenceManager* verifica si hubo algún error (*checkErrors*), en cuyo caso arroja una excepción al cliente (*PersistenceException* con código ENCODERERROR).

En el diagrama de la figura 7.9 se muestran las interacciones que se producen cuando una clase cliente solicita al *PersistenceManager* que lea un archivo.

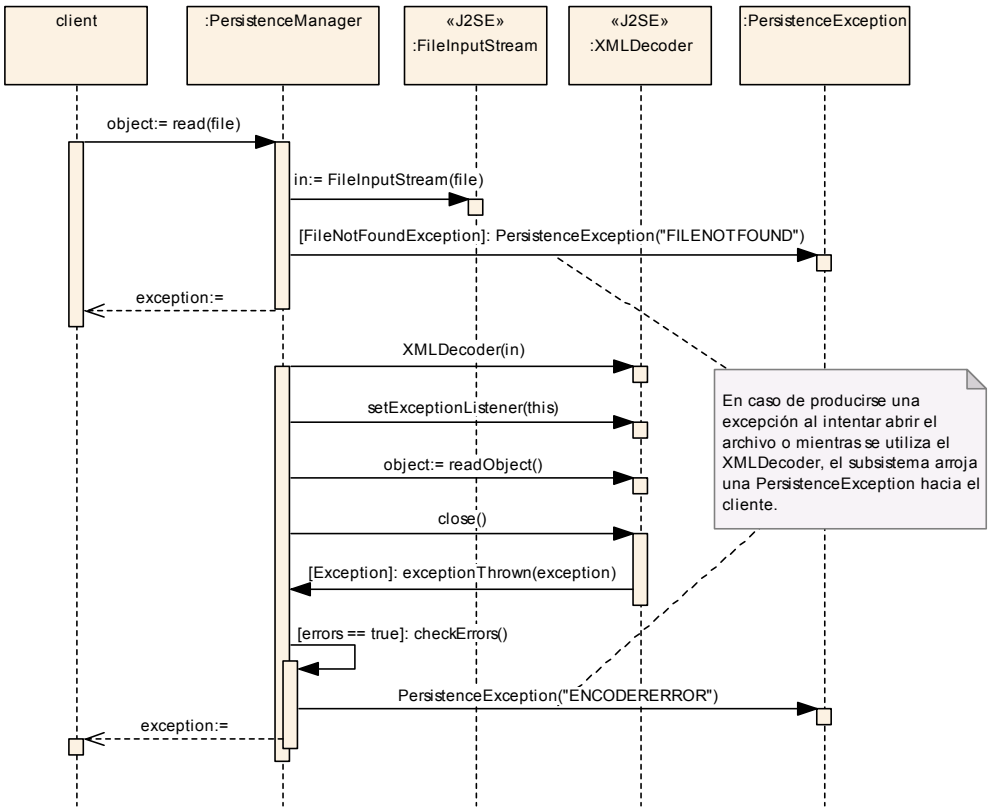

Figura 7.9. Lectura de un objeto desde archivo. La clase cliente envía el mensaje *read* al *PersistenceManager* pasándole como argumento el nombre del archivo fuente. El *PersistenceManager* crea un flujo de entrada (*FileInputStream*) utilizando las librerías de java. En caso de producirse una excepción (*FileNotFoundException*), instancia una *PersistenceException* con código de mensaje FILENOTFOUND y la arroja al cliente. En caso de no haber excepciones, el *PersistenceManager* crea una instancia de *XMLDecoder* pasándole como parámetro el flujo de entrada, y se registra como *listener* de excepciones del mismo. Finalmente, el *PersistenceManager* envía el mensaje *readObject* al *XMLDecoder* y lo cierra mediante el mensaje close.
En caso de producirse una excepción en alguno de los métodos del *XMLDecoder*, el mismo invoca al método *exceptionThrown* de su listener (el *PersistenceManager*), quien se encarga de setear el indicador de errores (atributo *errors*) en true. Finalmente, el *PersistenceManager* verifica si hubo algún error (*checkErrors*), en cuyo caso arroja una excepción al cliente (*PersistenceException* con código ENCODERERROR).

7.4.3 messages – Subsistema de manejo de mensajes

Este subsistema se encarga de la administración de los mensajes y etiquetas que se muestran al usuario. El mismo provee a las aplicaciones la capacidad de almacenar todos los mensajes y etiquetas en un archivo (archivo "messages"), lo que brinda la posibilidad de hacer que las mismas sean multiidioma (un archivo "messages" para cada idioma soportado por la aplicación). A continuación, se explica su estructura y su esquema de funcionamiento mediante un reporte extraído de la herramienta *Enterprise Architect*.

Reporte: messages - Estructura

Este subsistema contiene todas las clases encargadas del manejo de los mensajes que se muestran al usuario. El subsistema expone una interfaz mediante la cual se pueden recuperar mensajes de texto a partir de códigos.

El diagrama de la figura 7.10 muestra la estructura interna del subsistema.

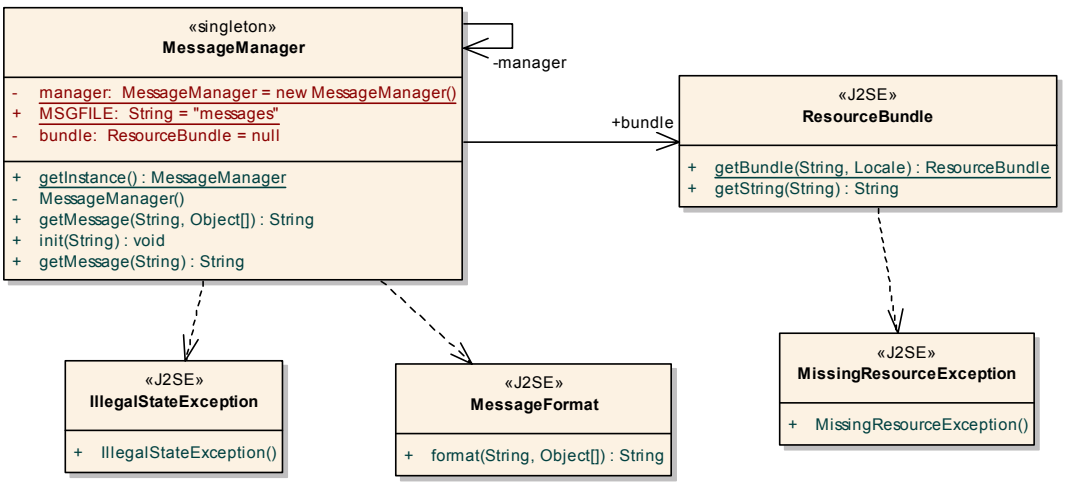

Figura 7.10. Clases del subsistema messages.

MessageManager

public «singleton» Class: Esta clase se encarga de la administración de etiquetas y mensajes de usuario dentro de la aplicación. Las clases de interfaz de y de negocios solicitan las etiquetas y los mensajes a esta clase. MessageManager encapsula la utilización de clases de la plataforma J2SE para simplificar el manejo de mensajes. La clase implementa el patrón de diseño "Singleton", lo que significa que existe un único objeto de esta clase en toda la aplicación.

MessageManager Attributes

Attribute	Type	Notes
manager	private static : *MessageManager*	Instancia retornada por el método getInstance. Es la única instancia de MessageManager de toda la aplicación (Singleton). Initial Value: new MessageManager();
MSGFILE	public static : *String*	Archivo de mensajes Initial Value: "messages";
bundle	private : *ResourceBundle*	ResourceBundle que encapsula el acceso al archivo de mensajes Initial Value: null;

MessageManager Methods

Method	Type	Notes
getInstance ()	public static: *MessageManager*	Retorna la única instancia de la clase.
MessageManager ()	private:	
getMessage (*String, Object[]*)	public: *String*	param: key [String - in] Clave identificatoria del mensaje. param: params [Object[] - in] Array de objetos a utilizar como parámetros del mensaje. Recupera un mensaje en el lenguaje solicitado por el cliente. El mensaje retorna como un String con el formato apropiado para su presentación.
init (*String*)	public: *void*	param: locale [String - in] Objeto que representa el idioma a utilizar en los mensajes. Permite inicializar la única instancia del MessageManager para un idioma determinado. Crea un objeto Locale y lo asigna al atributo locale.
getMessage (*String*)	public: *String*	param: key [String - in] Clave identificatoria del mensaje. Abreviatura de getMessage para recuperar mensajes sin argumentos

ResourceBundle

public «J2SE» Class: Clase encargada de la recuperación de mensajes desde archivos. Pertenece a las librerías de la plataforma J2SE.

ResourceBundle Methods

Method	Type	Notes
getBundle (*String, Locale*)	public static: *ResourceBundle*	param: file [String - in] Nombre del archivo con los mensajes de texto. param: locale [Locale - in] Objeto que representa el idioma. Se utiliza para localizar el archivo de mensajes. Permite obtener una instancia de la clase, cargada con los mensajes del archivo recibido como parámetro.
getString (*String*)	public: *String*	param: key [String - in] Clave que identifica al mensaje. Obtiene un mensaje a partir de una clave.

MessageFormat

public «J2SE» Class: Clase encargada de formatear mensajes de texto de acuerdo a un patrón base y a un conjunto de parámetros. Forma parte de las librerías de la plataforma J2SE.

MessageFormat Methods

Method	Type	Notes
format (*String, Object[]*)	public: *String*	param: pattern [String - in] Texto que brinda el recipiente para el mensaje formateado. param: params [Object[] - in] Array de objetos con los parámetros a incluir en el texto. Formatea un mensaje a partir de un texto y un conjunto de parámetros.

IllegalStateException

public «J2SE» Class: Excepción arrojada por el subsistema cuando se invoca al método getMessage y el subsistema aún no fue inicializado.

IllegalStateException Methods

Method	Type	Notes
IllegalStateException ()	public:	Constructor de default

MissingResourceException

public «J2SE» Class: Excepción arrojada por el ResourceBundle cuando no puede encontrar un string que le fue solicitado.

MissingResourceException Methods

Method	Type	Notes
MissingResourceException ()	public:	Constructor de default

Reporte: messages - Funcionamiento

Aquí se contemplan los aspectos de funcionamiento dinámico del subsistema de mensajes y etiquetas.

El funcionamiento se explica mediante diagramas de interacción donde se muestra la comunicación entre los objetos del subsistema en tiempo de ejecución.

En el diagrama de la figura 7.11 se muestran las interacciones que se producen cuando una clase cliente solicita al *MessageManager* un mensaje en particular.

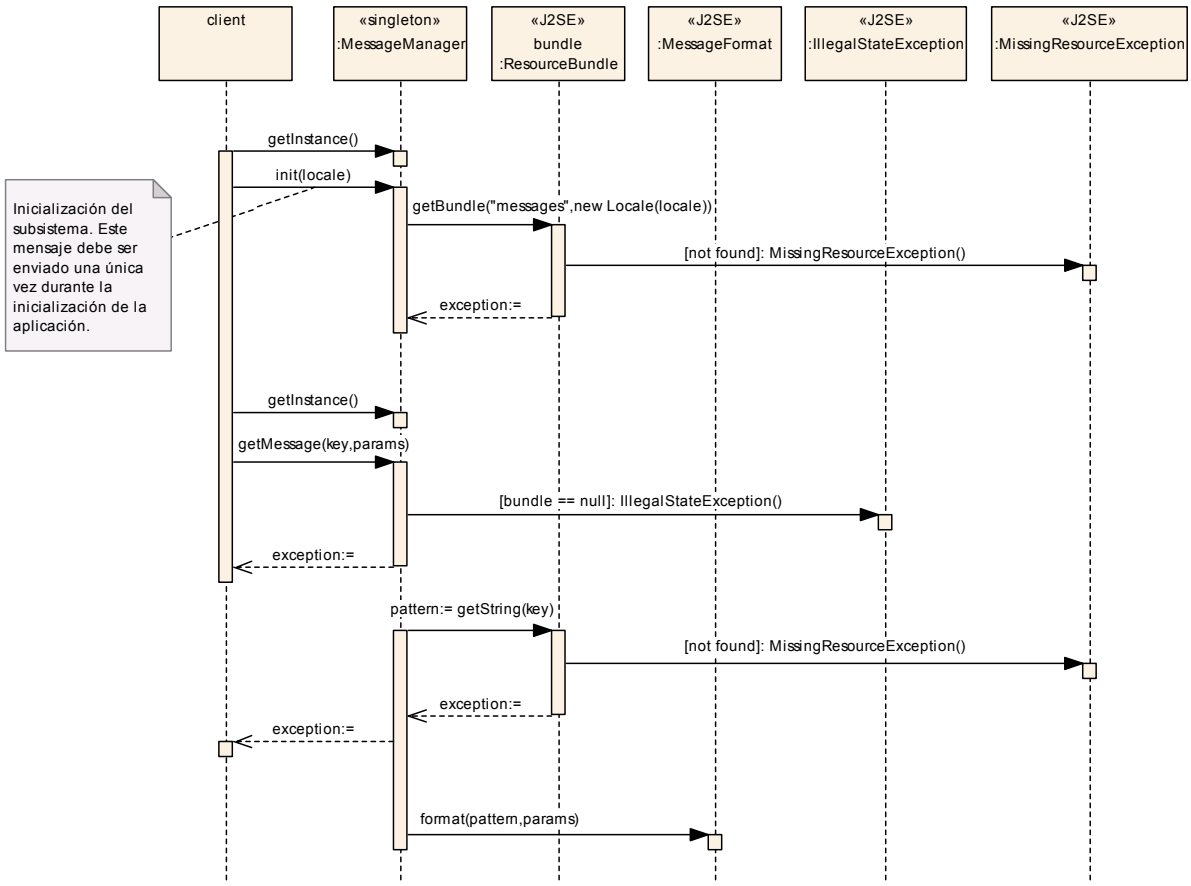

Figura 7.11. Inicio del subsistema y Recuperación de un mensaje. En la parte superior del diagrama se muestra la inicialización del subsistema. La misma consiste en que el código cliente (programa principal) obtiene la única instancia del *MessageManager* (método *getInstance*) y le envía el mensaje *init*, suministrándole como parámetro un string (*locale*) que representa el idioma utilizado por la aplicación. El *MessageManager* inicializa su objeto *ResourceBundle* asociado. Este objeto encapsula el acceso al archivo de mensajes (*messages_locale.properties*). En caso de no encontrar el archivo de mensajes, el *ResourceBundle* arroja una *MissingResourceException* hacia el *MessageManager*.
Una vez inicializado el subsistema, cuando un cliente quiere obtener un mensaje, se producen las siguientes interacciones. En primer lugar, el cliente accede a la única instancia del *MessageManager* mediante el método *getInstance*. A continuación le solicita el mensaje mediante el método *getMessage*, suministrándole como argumentos la clave identificatoria del mensaje y la lista de parámetros asociados al mismo (en caso de haberlos). En caso de no estar inicializado (bundle == null), el *MessageManager* arroja una *IllegalStateException* hacia el cliente. El *MessageManager* solicita el mensaje a partir de su clave al objeto *ResourceBundle* que tiene asociado, y finalmente utiliza la clase *MessageFormat* para darle formato antes de retornarlo al cliente. En caso de no encontrar el mensaje solicitado, el *ResourceBundle* arroja una *MissingResourceException*.

7.4.4 JasperReports – Subsistema de manejo de reportes

Este subsistema se encarga de la generación, previsualización e impresión de reportes. A continuación, se explica su estructura y esquema de funcionamiento mediante un reporte extraído de la herramienta *Enterprise Architect*.

Reporte: JasperReports - Estructura

Este subsistema representa al producto JasperReports utilizado para la generación de reportes. El mismo es una API que brinda un conjunto de clases utilizables directamente en la aplicación.

El diagrama de la figura 7.12 muestra su estructura interna.

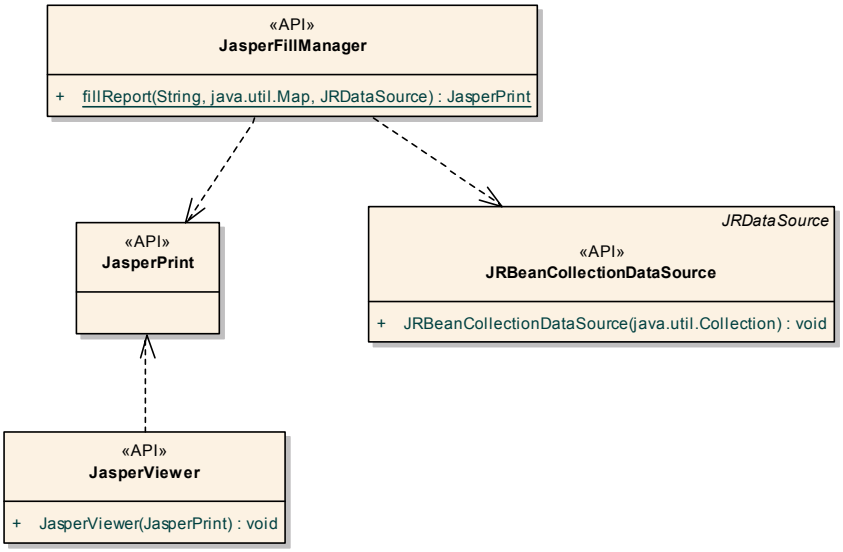

Figura 7.12. Clases del subsistema JasperReports.

JasperFillManager

public «API» Class: Esta clase permite llenar un reporte a partir de un conjunto de datos.

JasperFillManager Methods

Method	Type	Notes
fillReport (*String, java.util.Map, JRDataSource*)	public static: *JasperPrint*	param: report [String - in] Nombre de archivo con la plantilla del reporte param: params [java.util.Map - in] Parámetros a incluir en el reporte param: datasource [JRDataSource - in] Fuente con los datos del reporte Permite llenar un reporte con parámetros y datos

JasperPrint

public «API» Class: Representa la versión imprimible de un reporte.

JasperViewer

public «API» Class: Esta clase implementa la interfaz de previsualización de reportes. La interfaz permite imprimir el reporte visualizado y avanzar y retroceder por las hojas del mismo.

JasperViewer Methods

Method	Type	Notes
JasperViewer (*JasperPrint*)	public: *void*	param: report [JasperPrint - in] Versión imprimible de un reporte Constructor a partir de la versión imprimible de un reporte

JRBeanCollectionDataSource

public «API» Class

Implements: JRDataSource. : Esta clase representa la fuente de datos a utilizar en el reporte. Se utiliza inicializándola con una colección de objetos tipo JavaBean que contienen los datos del reporte.

JRBeanCollectionDataSource Methods

Method	Type	Notes
JRBeanCollectionDataSource (*java.util.Collection*)	public: *void*	param: list [java.util.Collection - in] Colección de JavaBeans Constructor a partir de una colección de JavaBeans

Reporte: JasperReports - Funcionamiento

Aquí se contemplan los aspectos de funcionamiento dinámico del subsistema de reportes. El funcionamiento se explica mediante diagramas de interacción donde se muestra la comunicación entre los objetos del subsistema en tiempo de ejecución.

En el diagrama de la figura 7.13 se muestran las interacciones que se producen para generar un reporte con las librerías de JasperReport.

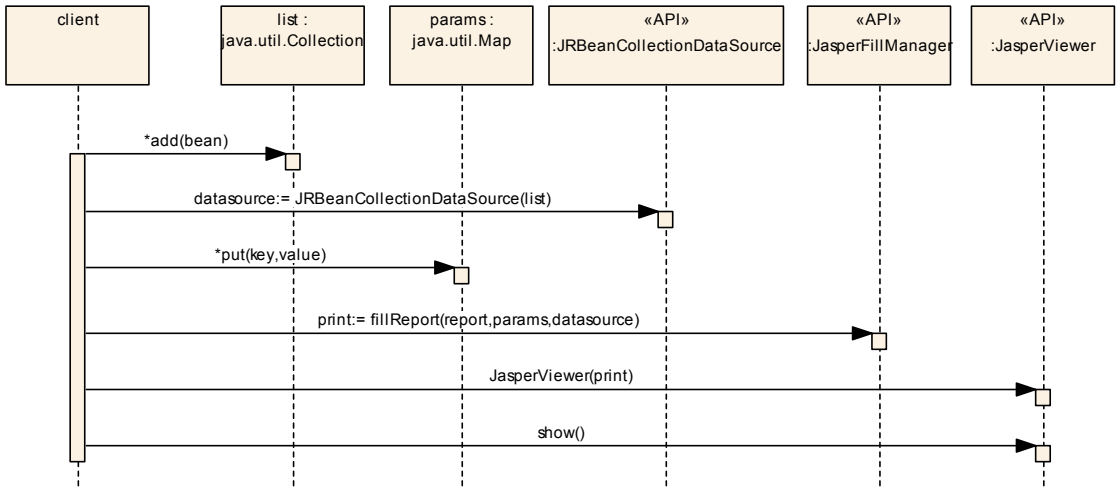

Figura 7.13. Generación de un reporte. En primer lugar, el código cliente (objeto client) llena una lista (java.util.Collection) con objetos tipo *JavaBean* (objetos que contienen solamente datos) que contienen los datos a mostrar en el reporte y provee la lista como argumento al constructor de *JRBeanCollectionDataSource*. Además de esos datos, también llena un mapa (java.util.Map) con parámetros genéricos del reporte (etiquetas a mostrar como por ejemplo el título o un encabezado). Finalmente, el código cliente invoca al método *fillReport* del *JasperFillManager* pasándole como argumentos el nombre del archivo con la plantilla del reporte, el mapa con los parámetros, y el datasource con la lista de datos. El método *fillReport* retorna un objeto *JasperPrint* (print en el diagrama). Por último, crea un objeto *JasperViewer* pasándole el objeto *JasperPrint* obtenido e invoca al método *show* para abrir la ventana de previsualización.

7.4.5 JavaHelp – Subsistema de manejo de ayudas

Este subsistema se encarga de la visualización de ayudas dentro de la aplicación. A continuación, se explica su estructura y esquema de funcionamiento mediante un reporte extraído de la herramienta *Enterprise Architect*.

Reporte: JavaHelp - Estructura

Este subsistema representa al producto JavaHelp utilizado para el manejo de ayudas dentro de la aplicación. El mismo es una API que brinda un conjunto de clases utilizables directamente en la aplicación.

El diagrama de la figura 7.14 muestra su estructura interna.

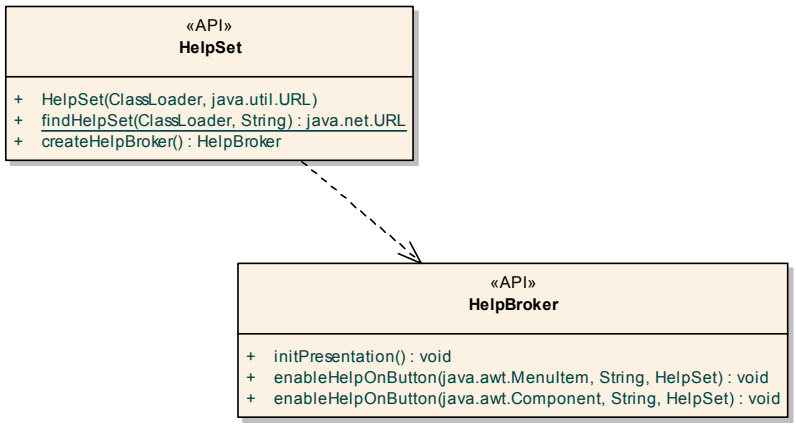

Figura 7.14. Clases del subsistema JavaHelp.

HelpBroker

public «API» Class: Esta clase se encarga de la habilitación de ayudas en los elementos de la interfaz de usuario, y de la presentación en general de los *HelpSets*.

HelpBroker Methods

Method	Type	Notes
initPresentation ()	public: *void*	Inicializa todos los aspectos de presentación de las ayudas
enableHelpOnButton (*java.awt.Component, String, HelpSet*)	public: *void*	param: component [java.awt.Component - in] Componente sobre el cual habilitar la ayuda param: id [String - in] Identificación de la ayuda a mostrar param: helpset [HelpSet - in] Referencia al HelpSet que contiene la ayuda a mostrar Habilita la ayuda sobre un botón u opción de menú, enlazándola con un identificador y un HelpSet
enableHelpOnButton (*java.awt.MenuItem, String, HelpSet*)	public: *void*	param: component [java.awt.MenuItem - in] Componente sobre el cual habilitar la ayuda param: id [String - in] Identificación de la ayuda a mostrar param: helpset [HelpSet - in] Referencia al HelpSet que contiene la ayuda a mostrar Habilita la ayuda sobre un botón u opción de menú, enlazándola con un identificador y un HelpSet

HelpSet

public «API» Class: Esta clase encapsula toda la información inherente al subsistema de ayudas (archivos xml con la definición de la Tabla de Contenidos,

Índices, y Mapa de Identificadores para asociar secciones de la ayuda con elementos específicos de las interfaces de usuario).

HelpSet Methods

Method	Type	Notes
HelpSet (*ClassLoader, java.util.URL*)	public:	param: loader [ClassLoader - in] Referencia a objeto ClassLoader (puede ser null) param: url [java.util.URL - in] Objeto URL para acceder al archivo de definiciones Constructor a partir de un ClassLoader y una URL para acceder al archivo de definiciones
findHelpSet (*ClassLoader, String*)	public static: *java.net.URL*	param: loader [ClassLoader - in] Referencia al objeto ClassLoader a utilizar para la carga del archivo param: name [String - in] Nombre del archivo de definición (helpset) Genera un objeto URL para el acceso a un archivo con las definiciones sobre las ayudas (archivo helpset)
createHelpBroker ()	public: *HelpBroker*	Crea un objeto HelpBroker

Reporte: JavaHelp - Funcionamiento

Aquí se contemplan los aspectos de funcionamiento dinámico del subsistema de ayudas.

El funcionamiento se explica mediante diagramas de interacción donde se muestra la comunicación entre los objetos del subsistema en tiempo de ejecución.

En el diagrama de la figura 7.15 se muestran las interacciones que se producen para habilitar las ayudas JavaHelp.

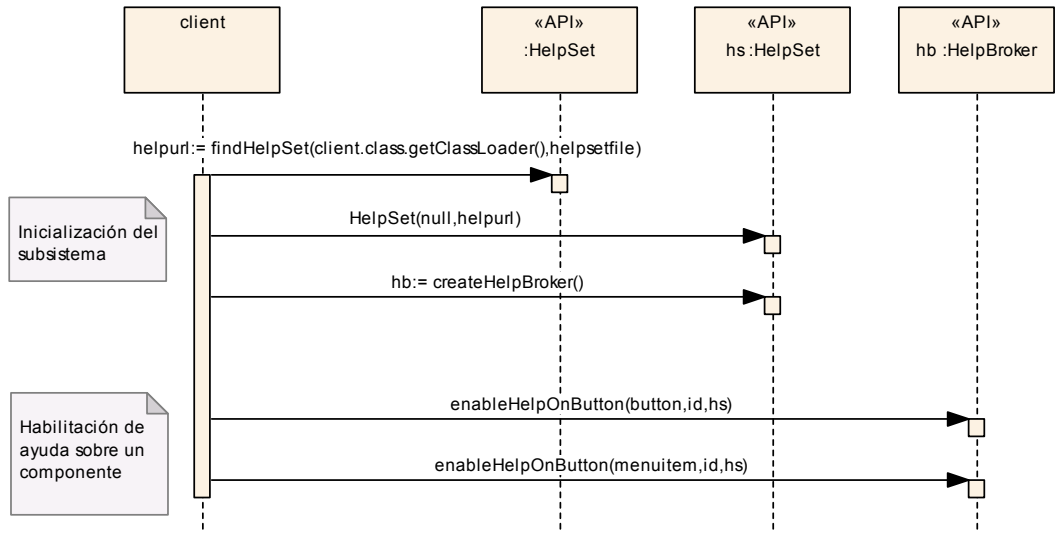

Figura 7.15. Activación de mecanismo de ayuda sobre un componente.

Todas las definiciones de la estructura de ayudas JavaHelp se encuentran en un archivo llamado *helpset*. En primer lugar, el código cliente (objeto client) obtiene un objeto URL de acceso al archivo *helpset* invocando al método *findHelpSet* de la clase *HelpSet*. Con la URL obtenida instancia un objeto *HelpSet* que representa toda la información de ayudas. A continuación crea un objeto *HelpBroker* invocando al método *createHelpBroker* del *HelpSet*. La habilitación de ayudas sobre botones o ítems de menú consiste simplemente en invocar al método *enableHelpOnButton* de la clase *HelpBroker*, pasándole como argumentos el objeto a habilitar (botón o ítem de menú), el identificador de la ayuda a mostrar (el cual mapea a un archivo html con la ayuda), y el objeto *HelpSet* obtenido en la inicialización.

7.5 Modelo de clases de diseño

El siguiente reporte extraído de *Enterprise Architect* muestra el modelo obtenido como resultado de las actividades "Diseño de los casos de uso reales" y "Diseño de clases" contempladas en la metodología. El reporte contiene las clases correspondientes a la capa específica del negocio (paquetes *views*, *controllers* y *models*). No contempla las clases correspondientes a los subsistemas y paquetes de soporte, ya que los mismos fueron explicados en el apartado anterior.

Reporte: Modelo de clases de diseño

views

Este paquete contiene todas las clases del tipo *view* (vista) correspondientes al patrón de diseño MVC. Estas clases brindan las interfaces gráficas de usuario y son derivadas de las clases del paquete Swing (J2SE).

Los diagramas de las figuras 7.16a y 7.16b muestran las clases del tipo *view* que conforman las interfaces gráficas de usuario.

Figura 7.16a. Clases del paquete views.

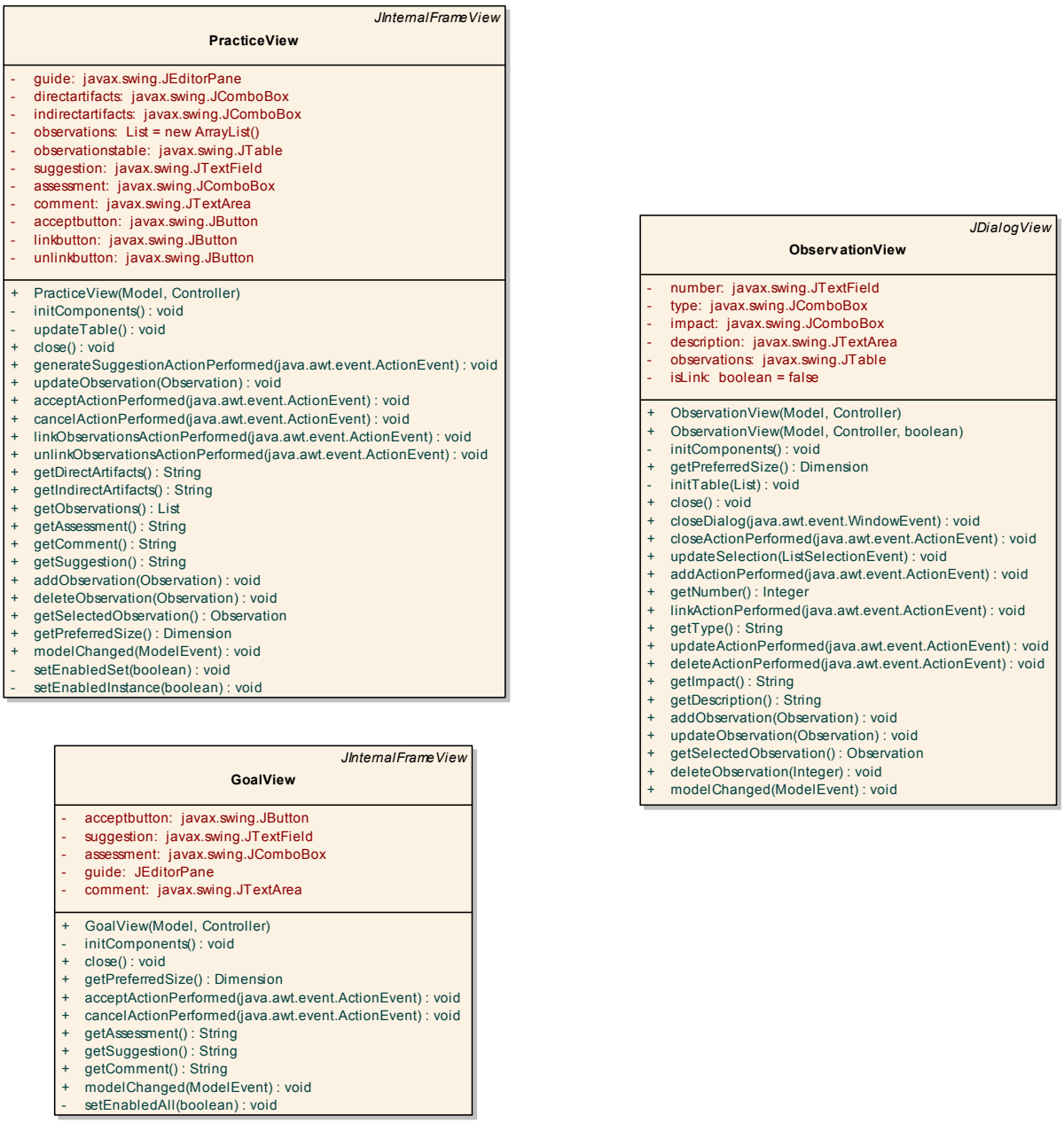

Figura 7.16b. Clases del paquete views.

El diagrama de la figura 7.17 muestra las relaciones de trazabilidad entre las clases de este paquete y las clases del modelo de análisis.

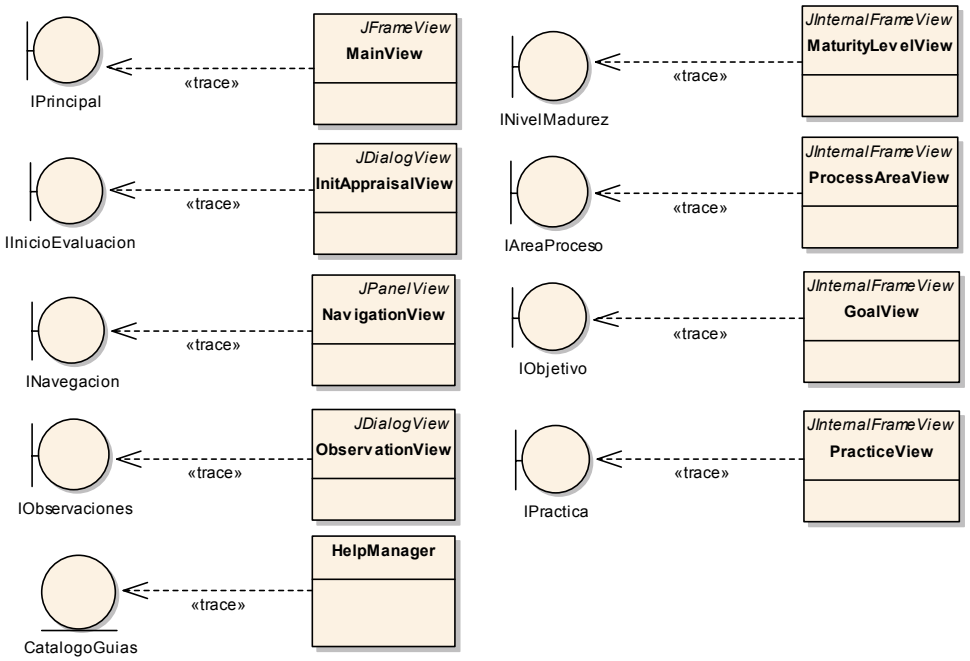

Figura 7.17. Trazabilidad.

MainView

public Class

Extends: JFrameView. : Representa la ventana principal del sistema. Contiene la barra de menú y el área inferior donde se muestra el árbol del modelo y los paneles de evaluación.

MainView Attributes

Attribute	Type	Notes
leftpanel	private : *JScrollPane*	Panel para la ventana de navegación
rightpanel	private : *JDesktopPane*	Panel para las ventanas de evaluación
editmenu	private : *javax.swing.JMenu*	Menú de edición
filemenu	private : *javax.swing.JMenu*	Menú de archivos
helpmenu	private : *javax.swing.JMenu*	Menú de ayuda
newappraisal	private : *javax.swing.JMenuItem*	Ítem de menú para nueva evaluación
observations	private : *javax.swing.JMenuItem*	Ítem de menú para administración de observaciones
openappraisal	private : *javax.swing.JMenuItem*	Ítem de menú para apertura de evaluaciones
saveappraisal	private : *javax.swing.JMenuItem*	Ítem de menú para almacenamiento de evaluaciones
generatereport	private : *javax.swing.JMenuItem*	Ítem de menú para generación de reporte
startassessment	private : *javax.swing.JMenuItem*	Ítem de menú para evaluación de elementos

MainView Methods

Method	Type	Notes
MainView (*Model, Controller*)	public:	param: model [Model - in] Modelo a asociar param: controller [Controller - in] Controlador a asociar Constructor a partir de un modelo y un controlador
initComponents ()	private: *void*	Inicializa la interfaz de usuario
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana principal
initDesktop ()	public: *void*	Inicializa el área de trabajo, que consiste en un panel dividido verticalmente. A la izquierda se ubicará la vista de navegación y a la derecha las distintas ventanas de evaluación.
exitActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa opción de menú Salir
exitForm (*java.awt.event.WindowEvent*)	private: *void*	param: evt [java.awt.event.WindowEvent - in] Procesa botón de cierre de aplicación en ventana principal
exit ()	private: *void*	Cierra el sistema
cleanDesktop ()	public: *void*	Limpia el área de trabajo
setLeftPane (*JPanelView*)	public: *void*	param: panel [JPanelView - in] Vista a ubicar en el panel izquierdo Setea el contenido del panel izquierdo de la ventana principal
newAppraisalActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Nueva evaluación.
addToRightPane (*JInternalFrame*)	public: *void*	param: frame [JInternalFrame - in] Vista a agregar Agrega una vista al panel derecho
saveAppraisalActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Guardar evaluación.
openAppraisalActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Abrir evaluación.
startAssessmentActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Evaluar
generateReportActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Generar reporte.

Method	Type	Notes
observationsActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Observaciones
aboutActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado a la opción de menú Acerca de...
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
main (*String[]*)	public static: *void*	param: args [String[] - in]

InitAppraisalView

public Class

Extends: JDialogView. : Representa la ventana de inicialización de evaluaciones. Contiene los datos sobre el objetivo de la evaluación, el alcance y la cantidad de instancias de evaluación.

InitAppraisalView Attributes

Attribute	Type	Notes
target	private : *JTextField*	Campo Objetivo
scope	private : *JComboBox*	Campo Alcance
instanceslist	private : *JList*	Campo Instancias
newinstance	private : *JTextField*	Campo para agregar instancias
addbutton	private : *JButton*	Botón Agregar
delbutton	private : *JButton*	Botón Eliminar

InitAppraisalView Methods

Method	Type	Notes
InitAppraisalView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociado.
initComponents ()	private: *void*	Inicializa la interfaz de usuario
close ()	public: *void*	Cierra la ventana
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana
closeDialog (*java.awt.event.WindowEvent*)	public: *void*	param: evt [java.awt.event.WindowEvent - in] Procesa el evento disparado por el botón de cierre de ventana
cancelActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Cancelar
getTarget ()	public: *String*	Obtiene el objetivo de la evaluación
acceptActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Aceptar.
getScope ()	public: *String*	Obtiene el alcance de la evaluación
getInstances ()	public: *List*	Obtiene la lista de instancias definidas para la evaluación
addInstanceActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Agregar.
scopeActionPerformed (*java.awt.event.ActionEvent*)	private: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por la combo de alcance.
delInstanceActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Eliminar.
main (*String*)	public static: *void*	param: args[] [String - in] Método para prueba unitaria de la interfaz

NavigationView

public Class

Extends: JPanelView. : Representa la interfaz de navegación del modelo CMMI-SW. Contiene un árbol en el que cada nodo es una parte del modelo.

NavigationView Attributes

Attribute	Type	Notes
jtree	private : *JTree*	Árbol con la representación del modelo
treemodel	private : *DefaultTreeModel*	Modelo asociado al árbol JTree
popup	private : *JPopupMenu*	Menú popup
NavigationController	public : *NavigationController*	
Appraisal	public : *Appraisal*	

NavigationView Methods

Method	Type	Notes
NavigationView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado a al vista param: controller [Controller - in] Controlador asociado a la vista Constructor que recibe como argumentos el modelo y el controlador asociados
initComponents ()	private: *void*	Inicializa la interfaz de usuario. Para ello barre el Composite del modelo (clase Appraisal con sus agregaciones) y construye un JTree utilizando los AbstractComponent como objetos que representan a los nodos. Solamente inicializa el primer nivel de nodos (nodos de nivel de madurez).
addChildren (*AppraisalComponent, DefaultMutableTreeNode*)	private: *void*	param: comp [AppraisalComponent - in] param: node [DefaultMutableTreeNode - in]
populateTree (*String*)	public: *void*	param: name [String - in] Nombre del nodo raíz Completa el JTree con los componentes del modelo a partir del nodo recibido como parámetro
updateModel ()	public: *void*	Actualiza el JTree cuando se producen cambios en el modelo asociado
findNode (*String*)	private: *DefaultMutableTreeNode*	param: name [String - in]
getSelectedElement ()	public: *AppraisalComponent*	Retorna el elemento seleccionado en el árbol del modelo
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
showPopup (*java.awt.event.MouseEvent*)	private: *void*	param: evt [java.awt.event.MouseEvent - in] Procesa el evento asociado al botón derecho del mouse sobre un elemento del árbol
	private: *void*	param: evt [java.awt.event.ActionEvent - in]

Method	Type	Notes
startAssessmentAction Performed (*java.awt.event.ActionE vent*)		Procesa el evento asociado a la opción de menú Evaluar
resetNode (*AppraisalComponent*)	public: *void*	param: node [AppraisalComponent - in] Nivel a resetear Resetea el nivel recibido como parámetro en el JTree con los componentes del modelo

MaturityLevelView

public Class

Extends: JInternalFrameView. : Representa la ventana para la evaluación del nivel de madurez.

MaturityLevelView Attributes

Attribute	Type	Notes
guide	private : *JEditorPane*	Guia online
suggestion	private : *JTextField*	Campo Valoración sugerida

MaturityLevelView Methods

Method	Type	Notes
MaturityLevelView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado a la vista param: controller [Controller - in] Controlador asociado a la vista Constructor que recibe el modelo y el controlador asociados a la vista
initComponents ()	private: *void*	Inicializa la interfaz de usuario.
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana principal
acceptActionPerformed (*java.awt.event.ActionE vent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Aceptar
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
close ()	public: *void*	Cierra la ventana
getSuggestion ()	public: *String*	Obtiene el valor del campo Valoración sugerida

ProcessAreaView

public Class

Extends: JInternalFrameView. : Representa la ventana para la evaluación de las áreas de proceso.

ProcessAreaView Attributes

Attribute	Type	Notes
guide	private : *JEditorPane*	Guía online
outofscope	private : *javax.swing.JCheckBox*	Campo Fuera de alcance
noevidence	private : *javax.swing.JCheckBox*	Campo Sin evidencia
suggestion	private : *javax.swing.JTextField*	Campo Valoración sugerida
assessment	private : *javax.swing.JComboBox*	Campo Valoración elegida
acceptbutton	private : *javax.swing.JButton*	Botón Aceptar
comment	private : *javax.swing.JTextArea*	Campo Justificación

ProcessAreaView Methods

Method	Type	Notes
ProcessAreaView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociados a la vista
initComponents ()	private: *void*	Inicializa la interfaz de usuario.
close ()	public: *void*	Cierra la ventana
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana principal
generateSuggestionActionPerformed (*java.awt.event.ItemEvent*)	public: *void*	param: evt [java.awt.event.ItemEvent - in] Procesa el evento disparado por la modificación de campos en al interfaz (campos Fuera de alcance y Sin evidencia objetiva)
acceptActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Aceptar
cancelActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Cancelar
getAssessment ()	public: *String*	Obtiene el valor del campo Valoración elegida
getOutofscope ()	public: *boolean*	Obtiene el valor del campo Fuera de alcance
getSuggestion ()	public: *String*	Obtiene el valor del campo Valoración sugerida
getComment ()	public: *String*	Obtiene el valor del campo Justificación
getNoevidence ()	public: *boolean*	Obtiene el valor del campo Sin evidencia
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
setEnabledAll (*boolean*)	private: *void*	param: value [boolean - in] Indicador de habilitación o inhabilitación Habilita o deshabilita los campos del formulario

GoalView

public Class

Extends: JInternalFrameView. : Representa la ventana para la evaluación de los objetivos.

GoalView Attributes

Attribute	Type	Notes
acceptbutton	private : *javax.swing.JButton*	Botón Aceptar
suggestion	private : *javax.swing.JTextField*	Campo Valoración sugerida
assessment	private : *javax.swing.JComboBox*	Campo Valoración elegida
guide	private : *JEditorPane*	Guía online
comment	private : *javax.swing.JTextArea*	Campo Justificación

GoalView Methods

Method	Type	Notes
GoalView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado a la vista param: controller [Controller - in] Controlador asociado a la vista Constructor a partir del modelo y el controlador asociados
initComponents ()	private: *void*	Inicializa la interfaz de usuario.
close ()	public: *void*	Cierra la ventana
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana principal
acceptActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Aceptar
cancelActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Cancelar
getAssessment ()	public: *String*	Obtiene el valor del campo Valoración elegida
getSuggestion ()	public: *String*	Obtiene el valor del campo Valoración sugerida
getComment ()	public: *String*	Obtiene el valor del campo Justificación
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
setEnabledAll (*boolean*)	private: *void*	param: value [boolean - in] Indicador de habilitación o inhabilitación Habilita o deshabilita los campos del formulario

PracticeView

public Class

Extends: JInternalFrameView. : Representa la ventana para la evaluación de las prácticas.

PracticeView Attributes

Attribute	Type	Notes
guide	private : *javax.swing.JEditorPane*	Guía online
directartifacts	private : *javax.swing.JComboBox*	Campo Artefactos directos
indirectartifacts	private : *javax.swing.JComboBox*	Campo Artefactos indirectos
observations	private : *List*	Lista de observaciones vinculaddas Initial Value: new ArrayList();
observationstable	private : *javax.swing.JTable*	Tabla de observaciones vinculadas
suggestion	private : *javax.swing.JTextField*	Campo Valoración sugerida
assessment	private : *javax.swing.JComboBox*	Campo Valoración elegida
comment	private : *javax.swing.JTextArea*	Campo Justificación
acceptbutton	private : *javax.swing.JButton*	Botón Aceptar
linkbutton	private : *javax.swing.JButton*	Botón Vincular
unlinkbutton	private : *javax.swing.JButton*	Botón Desvincular

PracticeView Methods

Method	Type	Notes
PracticeView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado a la vista param: controller [Controller - in] Controlador asociado a la vista Constructor que recibe el modelo y el controlador asociados a la vista
initComponents ()	private: *void*	Inicializa la interfaz de usuario.
updateTable ()	private: *void*	Inicializa la tabla de observaciones @param list Lista de observaciones a incluir en la tabla
close ()	public: *void*	Cierra la ventana
generateSuggestionActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por la modificación de campos en al interfaz

Method	Type	Notes
updateObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a actualizar Actualiza la observación recibida como parámetro en la tabla de observaciones
acceptActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Aceptar
cancelActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Cancelar
linkObservationsActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Vincular
unlinkObservationsActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento asociado al botón Desvincular
getDirectArtifacts ()	public: *String*	Obtiene el valor del campo Artefactos Directos
getIndirectArtifacts ()	public: *String*	Obtiene el valor del campo Artefactos indirectos
getObservations ()	public: *List*	Obtiene la lista de observaciones de la pantalla
getAssessment ()	public: *String*	Obtiene el valor del campo Valoración elegida
getComment ()	public: *String*	Obtiene el valor del campo Justificación
getSuggestion ()	public: *String*	Obtiene el valor del campo Valoración sugerida
addObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a agregar Agrega una observación a la tabla de observaciones
deleteObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a eliminar Elimina una observación actualizando la tabla de observaciones
getSelectedObservation ()	public: *Observation*	Obtiene la observación seleccionada en la pantalla
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana principal
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo
setEnabledSet (*boolean*)	private: *void*	param: value [boolean - in] Indicador de habilitación o inhabilitación Habilita o deshabilita los campos del formulario
setEnabledInstance (*boolean*)	private: *void*	param: value [boolean - in] Indicador de habilitación o inhabilitación Habilita o deshabilita los campos del formulario

ObservationView

public Class

Extends: JDialogView. : Representa la ventana de administración de observaciones. Contiene los datos sobre las observaciones identificadas.

ObservationView Attributes

Attribute	Type	Notes
number	private : *javax.swing.JTextField*	Campo Número
type	private : *javax.swing.JComboBox*	Campo Tipo
impact	private : *javax.swing.JComboBox*	Campo Impacto
description	private : *javax.swing.JTextArea*	Campo Descripción
observations	private : *javax.swing.JTable*	Tabla de observaciones existentes
isLink	private : *boolean*	Indicador de tipo de ventana (de Vinculación o de Administración) Por default es Administración Initial Value: false;

ObservationView Methods

Method	Type	Notes
ObservationView (*Model, Controller*)	public:	param: model [Model - in] Modelo asociado param: controller [Controller - in] Controlador asociado Constructor que recibe el modelo y el controlador asociado.
ObservationView (*Model, Controller, boolean*)	public:	param: model [Model - in] Modelo asociado a la vista param: controller [Controller - in] Controlador asociado a la vista param: isLink [boolean - in] Indicador de vinculación Constructor que recibe el modelo y el controlador asociado, y un flag que indica si se trata de una vinculación
initComponents ()	private: *void*	Inicializa la interfaz de usuario
getPreferredSize ()	public: *Dimension*	Setea el tamaño de la ventana
initTable (*List*)	private: *void*	param: list [List - in] Lista de observaciones a incluir en la tabla Inicializa la tabla de observaciones
close ()	public: *void*	Cierra la ventana
closeDialog (*java.awt.event.WindowEvent*)	public: *void*	param: evt [java.awt.event.WindowEvent - in] Procesa el evento disparado por el botón de cierre de ventana

Method	Type	Notes
closeActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Cancelar
updateSelection (*ListSelectionEvent*)	public: *void*	param: evt [ListSelectionEvent - in]
addActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Agregar
getNumber ()	public: *Integer*	Obtiene el campo Número de observación seleccionada
linkActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Vincular
getType ()	public: *String*	Obtiene el campo Tipo de la observación seleccionada
updateActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Modificar
deleteActionPerformed (*java.awt.event.ActionEvent*)	public: *void*	param: evt [java.awt.event.ActionEvent - in] Procesa el evento disparado por el botón Eliminar
getImpact ()	public: *String*	Obtiene el campo Impacto de la observación seleccionada
getDescription ()	public: *String*	Obtiene el campo Descripción de la observación seleccionada
addObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a agregar Agrega una observación a la tabla de observaciones
updateObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a actualizar Actualiza la observación recibida como parámetro en la tabla de observaciones
getSelectedObservation ()	public: *Observation*	Retorna las observaciones seleccionadas en la pantalla
deleteObservation (*Integer*)	public: *void*	param: number [Integer - in] Número identificatorio de la observación a eliminar Elimina la observación recibida como parámetro de la tabla de observaciones
modelChanged (*ModelEvent*)	public: *void*	param: evt [ModelEvent - in] Procesa las notificaciones de cambios en el modelo

ComboObject

public Class: Esta clase representa los objetos utilizados para llenar las combo boxes Cada elemento de una combo está compuesto por un código y un nombre El nombre se muestra al usuario y el código se utiliza internamente en la aplicación

ComboObject Attributes

Attribute	Type	Notes
code	private : *String*	
name	private : *String*	

ComboObject Methods

Method	Type	Notes
ComboObject (*String, String*)	public:	param: code [String - in] Código del elemento param: name [String - in] Nombre a mostrar al usuario Constructor a partir de un código y un nombre
getCode ()	public: *String*	Recupera el código
toString ()	public: *String*	Representación como String del objeto. Toma en cuenta solamente el nombre.

HelpManager

public Class: Esta clase se encarga de la administración de ayudas y guías online dentro de la aplicación

HelpManager Attributes

Attribute	Type	Notes
manager	private static : *HelpManager*	Instancia retornada por el método getInstance. Es la única instancia de HelpManager de toda la aplicación (Singleton). Initial Value: new HelpManager();
hb	private : *HelpBroker*	Referencia al HelpBroker de JavaHelp Initial Value: null;
hs	private : *HelpSet*	Referencia al HelpSet de JavaHelp Initial Value: null;
locale	private : *String*	Locale utilizado en la aplicación Initial Value: null;

HelpManager Methods

Method	Type	Notes
HelpManager ()	private:	Constructor privado de default
getInstance ()	public static: *HelpManager*	Retorna la única instancia de la clase.
enableHelp (*java.awt.Component, String*)	public: *void*	param: component [java.awt.Component - in] Elemento a habilitar param: id [String - in] Identificador de ayuda a asignar Habilita la ayuda para un botón
enableHelp (*java.awt.MenuItem, String*)	public: *void*	param: component [java.awt.MenuItem - in] Elemento a habilitar param: id [String - in] Identificador de ayuda a asignar Habilita la ayuda para un ítem de menú

init (*String*)	public: *void*	param: locale [String - in] Objeto que representa el idioma a utilizar en las ayudas. Permite inicializar la única instancia del HelpManager para un idioma determinado.
getGuide (*AppraisalComponent*)	public: *URL*	param: component [AppraisalComponent - in] Elemento asociado a la guía Obtiene la URL de cada guía en base al elemento indicado como parámetro (nivel, área de proceso, objetivo, práctica)

controllers

Este paquete contiene todas las clases del tipo *controller* (controlador) correspondientes al patrón de diseño MVC. Estas clases procesan las acciones efectuadas por el usuario en las *views* y las traducen a eventos o invocaciones en el *model* (lógica de negocios de la aplicación).

El diagrama de la figura 7.18 muestra las clases del tipo *controller* encargadas del procesamiento de las acciones que hace el usuario en las *views*. El procesamiento consiste en recuperar los datos provistos por el usuario y traducirlos a invocaciones en el *model*.

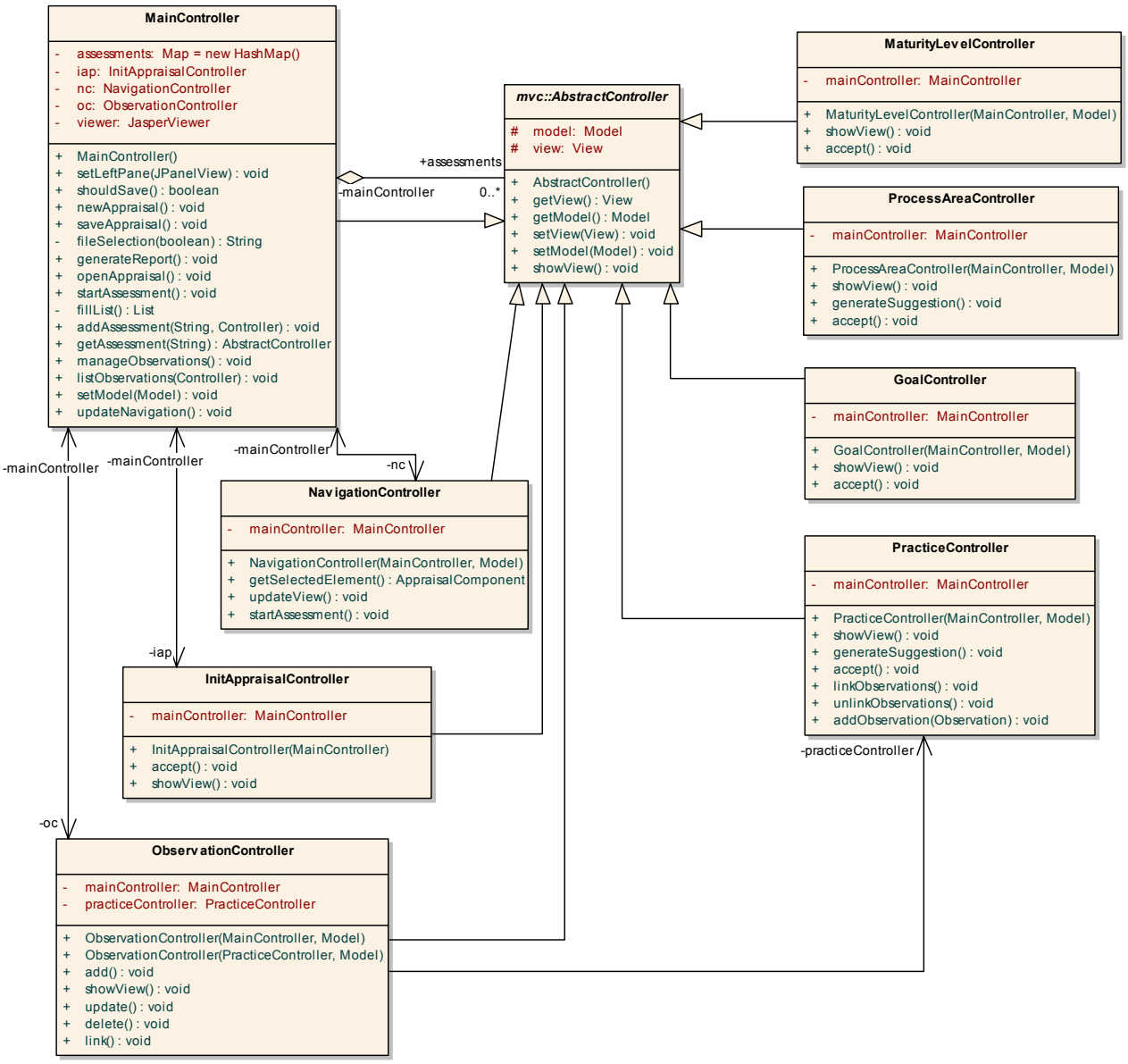

Figura 7.18. Clases del paquete controllers.

El diagrama de la figura 7.19 muestra las relaciones de trazabilidad existentes entre las clases de este paquete y las clases del modelo de análisis.

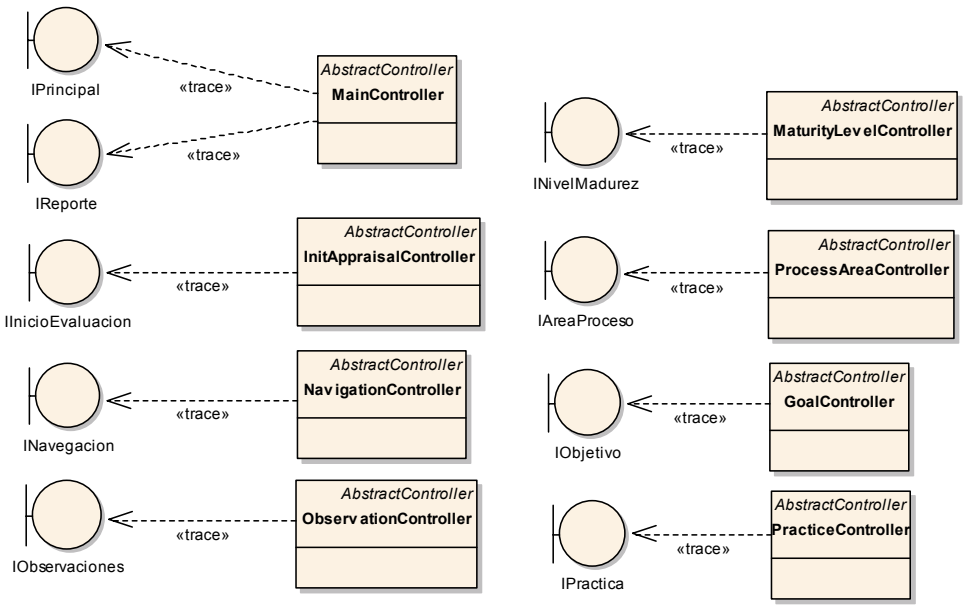

Figura 7.19. Trazabilidad.

MainController

public Class

Extends: AbstractController. : Controlador de la ventana principal de la aplicación. Se encarga de procesar las acciones que realiza el usuario en la ventana principal (principalmente acciones de menú).

MainController Attributes

Attribute	Type	Notes
assessments	private : *Map*	Lista de evaluaciones en curso. Cada entrada contiene un controlador de ventana de evaluación. Initial Value: new HashMap();
iap	private : *InitAppraisalController*	Referencia al InitAppraisalController
nc	private : *NavigationController*	Referencia al InitAppraisalController
oc	private : *ObservationController*	Referencia al ObservationController
viewer	private : *JasperViewer*	Referencia al viewer de Jasper

MainController Methods

Method	Type	Notes
MainController ()	public:	Constructor de default
setLeftPane (*JPanelView*)	public: *void*	param: panel [JPanelView - in] Vista a ubicar en el panel izquierdo Setea el contenido del panel izquierdo de la ventana principal

Method	Type	Notes
shouldSave ()	public: *boolean*	Pregunta al usuario si desea salvar una evaluación en curso
newAppraisal ()	public: *void*	Procesa la acción de crear una nueva evaluación
saveAppraisal ()	public: *void*	Procesa la acción de almacenar una evaluación. Presenta un JFileChooser al usuario para que indique el nombre del archivo de destino.
fileSelection (*boolean*)	private: *String*	param: open [boolean - in] Indicador de recuperación o almacenamiento de archivo Muestra ventana para seleccionar archivos @return Nombre del archivo seleccionado
generateReport ()	public: *void*	Procesa la acción de generar un reporte. Genera el reporte utilizando la API de JasperReports y lo muestra al usuario.
openAppraisal ()	public: *void*	Procesa la acción de recuperar una evaluación. Presenta un JFileChooser al usuario para que indique el nombre del archivo fuente.
fillList ()	private: *List*	Llena una lista de ElementBeans con los resultados de las áreas de proceso
startAssessment ()	public: *void*	Procesa la acción de evaluar un elemento
addAssessment (*String, Controller*)	public: *void*	param: name [String - in] Nombre del elemento bajo evaluación param: controller [Controller - in] Controlador que maneja la ventana de evaluación Agrega un controlador de evaluación a la lista de evaluaciones en curso
getAssessment (*String*)	public: *AbstractControll er*	param: name [String - in] Nombre del elemento bajo evaluación Obtiene un controlador de evaluación de la lista de evaluaciones en curso
manageObservations ()	public: *void*	Procesa la acción de administrar observaciones
listObservations (*Controller*)	public: *void*	param: controller [Controller - in] Controlador que invoca la acción de vincular observaciones Procesa la acción de abrir la ventana para vincular observaciones
setModel (*Model*)	public: *void*	param: newVal [Model - in] Modelo a setear Setea el modelo asociado al controlador. Propaga el modelo a la vista y le da la orden de inicializar el desktop.
updateNavigation ()	public: *void*	Actualiza la ventana de navegación del modelo

InitAppraisalController

public Class

Extends: AbstractController. : Controlador de la ventana de inicialización de evaluaciones. Se encarga de procesar las acciones que realiza el usuario en la ventana.

InitAppraisalController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

InitAppraisalController Methods

Method	Type	Notes
InitAppraisalController (*MainController*)	public:	param: mainController [MainController - in] Referencia al controlador principal Constructor que recibe una referencia el controlador principal.
accept ()	public: *void*	Procesa la acción de Aceptar el inicio de la evaluación
showView ()	public: *void*	Muestra la vista asociada

NavigationController

public Class

Extends: AbstractController. : Controlador de la ventana de navegación del modelo. Se encarga de procesar las acciones que realiza el usuario en el árbol del modelo.

NavigationController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

NavigationController Methods

Method	Type	Notes
NavigationController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al controlador principal param: model [Model - in] Modelo asociado al controlador Constructor que recibe como parámetros una referencia al MainController y una referencia al modelo asociado @param main_controller Referencia al controlador principal
getSelectedElement ()	public: *AppraisalComponent*	Retorna el elemento seleccionado en el árbol del modelo
updateView ()	public: *void*	Actualiza la vista de navegación del modelo
startAssessment ()	public: *void*	Procesa la acción de evaluar un elemento

MaturityLevelController

public Class

Extends: AbstractController. : Controlador de la ventana de evaluación de nivel de madurez. Se encarga de procesar las acciones que realiza el usuario en esa ventana.

MaturityLevelController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

MaturityLevelController Methods

Method	Type	Notes
MaturityLevelController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al MainController param: model [Model - in] Referencia al Model Constructor que recibe como argumentos una referencia al MainController y una referencia al Modelo
showView ()	public: *void*	Muestra la vista asociada
accept ()	public: *void*	Procesa la acción de Aceptar en la ventana de evaluación de nivel de madurez

ProcessAreaController

public Class

Extends: AbstractController. : Controlador de la ventana de evaluación de áreas de proceso. Se encarga de procesar las acciones que realiza el usuario en esa ventana.

ProcessAreaController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

ProcessAreaController Methods

Method	Type	Notes
ProcessAreaController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al MainController param: model [Model - in] Referencia al Model Constructor que recibe como argumentos una referencia al MainController y una referencia al modelo @param main_controller Referencia al MainController
showView ()	public: *void*	Muestra la vista asociada
generateSuggestion ()	public: *void*	Procesa la acción de modificar campos que afectan a la valoración sugerida
accept ()	public: *void*	Procesa la acción de Aceptar en la ventana de evaluación de áreas de proceso

GoalController

public Class

Extends: AbstractController. : Controlador de la ventana de evaluación de objetivos. Se encarga de procesar las acciones que realiza el usuario en esa ventana.

GoalController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

GoalController Methods

Method	Type	Notes
GoalController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al MainController param: model [Model - in] Referencia al Model Constructor que recibe como argumentos una referencia al MainController y una referencia al modelo @param main_controller Referencia al MainController
showView ()	public: *void*	Pone en primer plano la vista asociada al controlador
accept ()	public: *void*	Procesa la acción de Aceptar en la ventana de evaluación de objetivos

PracticeController

public Class

Extends: AbstractController. : Controlador de la ventana de evaluación de prácticas. Se encarga de procesar las acciones que realiza el usuario en esa ventana.

PracticeController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController

PracticeController Methods

Method	Type	Notes
PracticeController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al MainController param: model [Model - in] Modelo asociado Constructor que recibe una referencia al MainController y otra al modelo asociado @param main_controller Referencia al MainController
showView ()	public: *void*	Pone en primer plano la vista asociada al controlador
generateSuggestion ()	public: *void*	Procesa la acción de modificar campos que afectan a la valoración sugerida
accept ()	public: *void*	Procesa la acción de Aceptar en la ventana de evaluación de prácticas
linkObservations ()	public: *void*	Procesa la acción de vincular observaciones
unlinkObservations ()	public: *void*	Procesa la acción de desvincular observaciones
addObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observacion a agregar Agrega una observación a la lista de la pantalla

ObservationController

public Class

Extends: AbstractController. : Controlador de la ventana de administración de observaciones. Se encarga de procesar las acciones que realiza el usuario en la ventana.

ObservationController Attributes

Attribute	Type	Notes
mainController	private : *MainController*	Referencia al MainController
practiceController	private : *PracticeController*	Referencia al PracticeController

ObservationController Methods

Method	Type	Notes
ObservationController (*MainController, Model*)	public:	param: mainController [MainController - in] Referencia al MainController param: model [Model - in] Referencia al modelo Constructor que recibe una referencia al MainController y al Appraisal @param main_controller Referencia al MainController
ObservationController (*PracticeController, Model*)	public:	param: practiceController [PracticeController - in] Referencia al PracticeController que invoca param: model [Model - in] Modelo asociado al controlador Constructor que recibe una referencia a un PracticeController y al Appraisal @param practice_controller Referencia al PracticeController que invoca
add ()	public: *void*	Procesa la acción de agregar una nueva observación
showView ()	public: *void*	Pone en primer plano la vista asociada al controlador
update ()	public: *void*	Procesa la acción de modificar una observación
delete ()	public: *void*	Procesa la acción de eliminar una observación
link ()	public: *void*	Procesa la acción de vincular observaciones

models

Este paquete contiene todas las clases del tipo *model* (modelo) correspondientes al patrón de diseño MVC. Estas clases brindan toda la funcionalidad específica del negocio, y constituyen en sí mismas la lógica de negocios de la aplicación.

El diagrama de la figura 7.20 muestra las entidades de negocios de la aplicación.

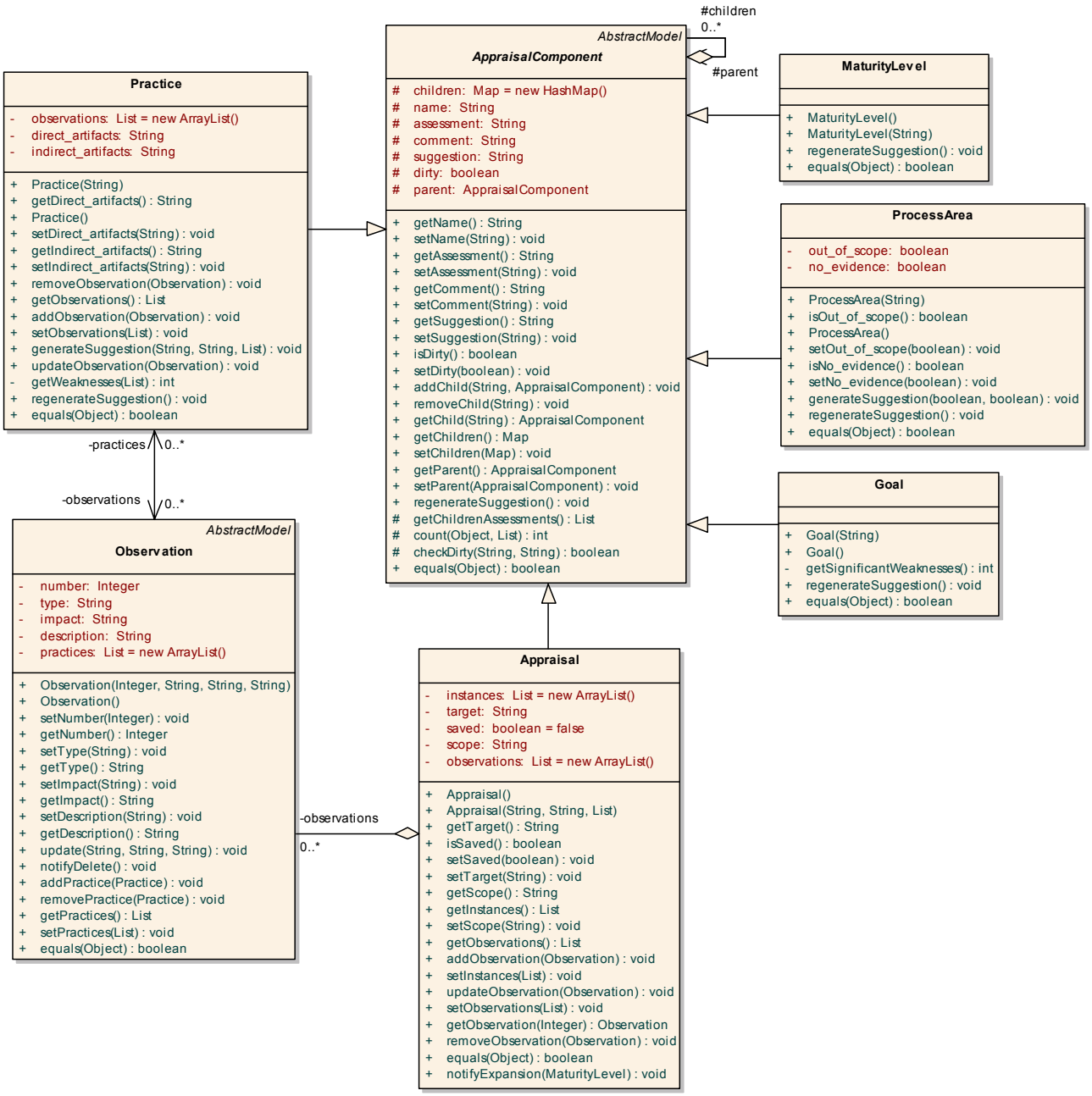

Figura 7.20. Entidades del paquete models.

El diagrama de la figura 7.21 muestra las clases de control encargadas de coordinar a las entidades de negocio de la aplicación.

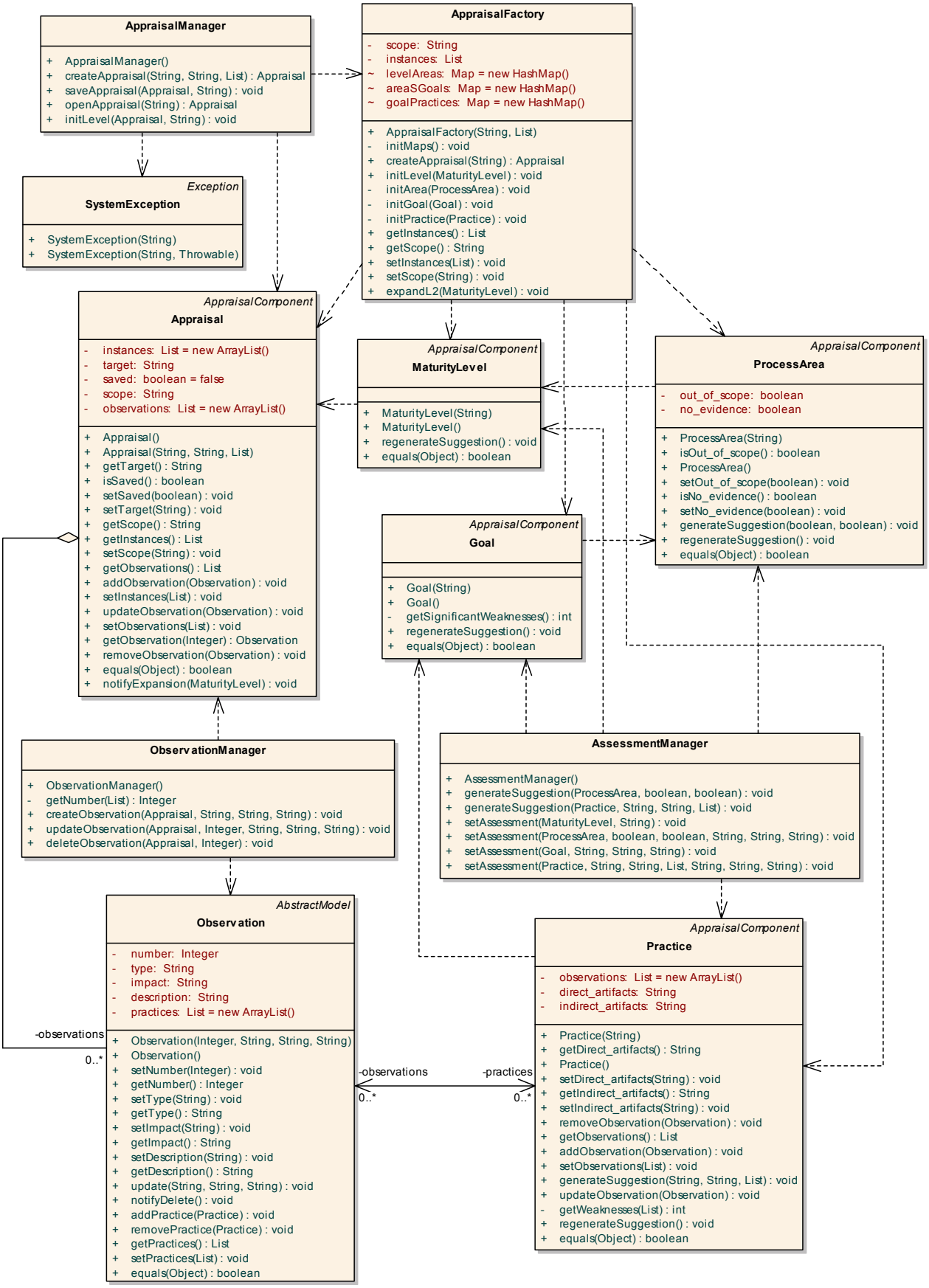

Figura 7.21. Controladores del paquete models.

El diagrama de la figura 7.22 muestra las relaciones de trazabilidad de las clases de este paquete con las clases del modelo de análisis.

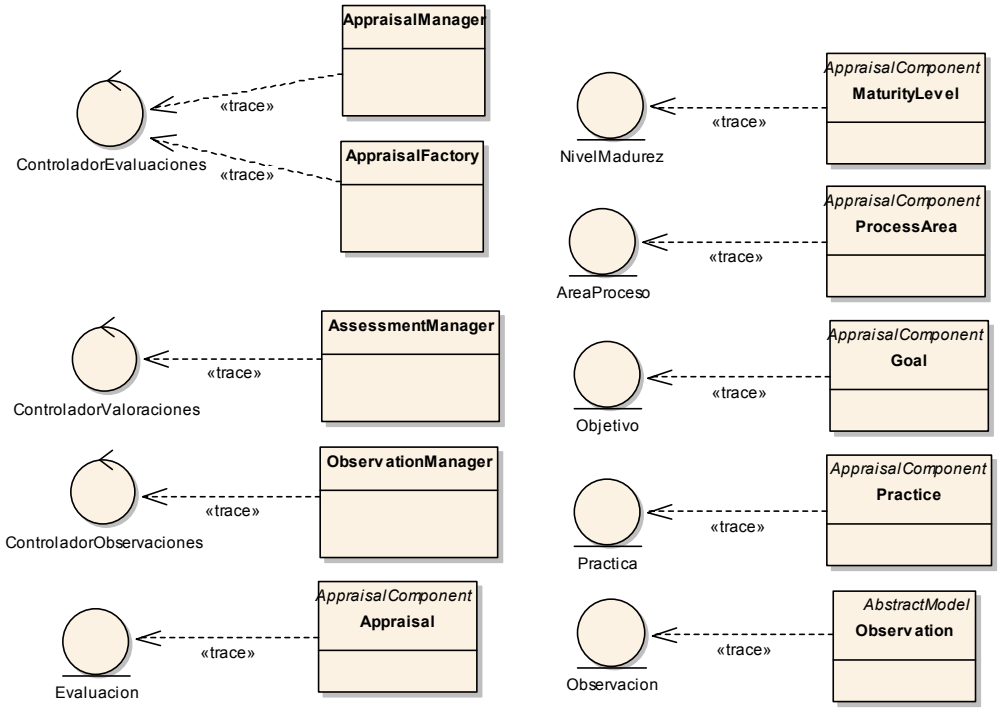

Figura 7.22. Trazabilidad.

AppraisalComponent

public abstract Class

Extends: AbstractModel. : Clase abstracta que representa la interface común provista por todos los componentes de una evaluación. Es subclase de *AbstractModel*. Implementa el patrón de diseño "*Composite*".

AppraisalComponent Attributes

Attribute	Type	Notes
children	protected : *Map*	Hijos del componente Initial Value: new HashMap();
name	protected : *String*	Nombre del componente
assessment	protected : *String*	Valoración actual del componente
comment	protected : *String*	Comentario asociado a la valoración actual
suggestion	protected : *String*	Valoración sugerida en base a las reglas de SCAMPI
dirty	protected : *boolean*	Indica si el componente ha cambiado su valoración sugerida y necesita revisión
parent	protected : *AppraisalComponent*	Componente padre

AppraisalComponent Methods

Method	Type	Notes
getName ()	«property get» public: *String*	Retorna el nombre del componente
setName (*String*)	«property set» public: *void*	param: newVal [String - in] Nombre a setear Setea el nombre del componente
getAssessment ()	«property get» public: *String*	Obtiene la valoración del componente
setAssessment (*String*)	«property set» public: *void*	param: new_assessment [String - in] Valoración a setear Setea la valoración del componente. En caso de ser diferente a la valoración actual, notifica al padre para que regenere su valoración sugerida
getComment ()	«property get» public: *String*	Obtiene el comentario asociado a la valoración actual
setComment (*String*)	«property set» public: *void*	param: newVal [String - in] Comentario a setear Setea el comentario asociado a la valoración actual
getSuggestion ()	«property get» public: *String*	Obtiene la valoración sugerida de acuerdo a las reglas de SCAMPI
setSuggestion (*String*)	«property set» public: *void*	param: newVal [String - in] Valoración a setear Setea la valoración sugerida
isDirty ()	«property get» public: *boolean*	Obtiene el indicador de necesidad de revisión
setDirty (*boolean*)	«property set» public: *void*	param: newVal [boolean - in] Valor a setear Setea el indicador de necesidad de revisión
addChild (*String, AppraisalComponent*)	public: *void*	param: name [String - in] Nombre del hijo a agregar param: child [AppraisalComponent - in] Hijo a agregar. Permite agregar un hijo al un composite.
removeChild (*String*)	public: *void*	param: name [String - in] Nombre del hijo a eliminar Permite eliminar un hijo de un composite
getChild (*String*)	public: *AppraisalComponent*	param: name [String - in] Nombre del hijo a obtener Obtiene un hijo de un composite
getChildren ()	public: *Map*	Obtiene todos los hijos de un componente
setChildren (*Map*)	public: *void*	param: newVal [Map - in] Mapa de hijos a setear Setea todos los hijos del componente @param children Mapa de hijos a setear

Method	Type	Notes
getParent ()	«property get» public: *AppraisalComponent*	Obtiene el padre del componente
setParent (*AppraisalComponent*)	«property set» public: *void*	param: newVal [AppraisalComponent - in] Padre a setear Setea el padre del componente
regenerateSuggestion ()	public: *void*	Recalcula la valoración sugerida debido a cambios producidos en los componentes inferiores del modelo
getChildrenAssessments ()	protected: *List*	Obtiene una lista con las valoraciones de los hijos En caso de no haber hijos, retorna una lista vacía
count (*Object, List*)	protected: *int*	param: object [Object - in] Objeto a buscar param: list [List - in] Lista de valoraciones Obtiene la cantidad de ocurrencias de un objeto en una lista
checkDirty (*String, String*)	protected: *boolean*	param: old_suggestion [String - in] Valoración sugerida anterior param: new_suggestion [String - in] Valoración sugerida actual Chequea si es necesario setear la indicación de necesidad de revisión en función de los cambios en las valoraciones sugeridas @return true en caso de necesidad de setear la indicación, false en caso contrario
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

Appraisal

public Class

Extends: AppraisalComponent. : Representa una evaluación de CMMI-SW aplicada a una organización o proyecto.

Appraisal Attributes

Attribute	Type	Notes
instances	private : *List*	Instancias a evaluar Initial Value: new ArrayList();
PRACTICES	public const static : *String*	Constante para definición de alcance a nivel de prácticas Initial Value: "PRACTICES";
target	private : *String*	Objetivo de la evaluación
GOALS	public const static : *String*	Constante para definición de alcance a nivel de objetivos Initial Value: "GOALS";
saved	private : *boolean*	Indica si la evaluación fue almacenada o necesita almacenarse Initial Value: false;

Attribute	Type	Notes
scope	private : *String*	Alcance de la evaluación
observations	private : *List*	Observaciones identificadas en la evaluación Initial Value: new ArrayList();
PROCESSAREAS	public const static : *String*	Constante para definición de alcance a nivel de áreas de proceso Initial Value: "PROCESSAREAS";

Appraisal Methods

Method	Type	Notes
Appraisal ()	public:	Constructor de default
Appraisal (*String, String, List*)	public:	param: target [String - in] Objetivo de la evaluación param: scope [String - in] Alcance de la evaluación param: instances [List - in] Instancias de evaluación Constructor a partir del objetivo, el alcance y las instancias de evaluación
getTarget ()	public: *String*	Retorna el objetivo de la evaluación
isSaved ()	«property get» public: *boolean*	Obtiene el valor del atributo saved
setSaved (*boolean*)	«property set» public: *void*	param: newVal [boolean - in] Valor a setear Setea el valor del atributo saved Notifica a los interesados con un ModelEvent tipo "Appraisal Saved"
setTarget (*String*)	public: *void*	param: newVal [String - in] Objetivo a setear Setea el objetivo de la evaluación
getScope ()	public: *String*	Retorna el alcance de la evaluación
getInstances ()	public: *List*	Retorna las instancias a evaluar
setScope (*String*)	public: *void*	param: newVal [String - in] Alcance a setear Setea el alcance de la evaluación
getObservations ()	public: *List*	Retorna las observaciones contenidas en la evaluación
addObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a agregar Agrega una observación a la lista de observaciones
setInstances (*List*)	public: *void*	param: list [List - in] Lista de instancias a setear Setea las instancias a evaluar
updateObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a actualizar Modifica una observación de la lista de observaciones

Method	Type	Notes
setObservations (*List*)	public: *void*	param: list [List - in] Lista de observaciones a setear Setea las observaciones contenidas en la evaluación
getObservation (*Integer*)	public: *Observation*	param: number [Integer - in] Número identificatorio de la observación a retornar Retorna una observación de la lista de observaciones
removeObservation (*Observation*)	public: *void*	param: obs [Observation - in] Referencia a la observación a eliminar Elimina una observación de la lista de observaciones
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals
notifyExpansion (*MaturityLevel*)	public: *void*	param: level [MaturityLevel - in] Nivel expandido Notifica a los interesados con un ModelEvent tipo "Child Modified"

MaturityLevel

public Class

Extends: AppraisalComponent. : Representa un nivel de madurez del modelo CMMI-SW.

MaturityLevel Attributes

Attribute	Type	Notes
REACHED	public const static : *String*	Constante para la valoración "Alcanzado" Initial Value: "REACHED";
NOT_REACHED	public const static : *String*	Constante para la valoración "No alcanzado" Initial Value: "NOT_REACHED";

MaturityLevel Methods

Method	Type	Notes
MaturityLevel (*String*)	public:	param: name [String - in] Nombre del nivel Constructor a partir del nombre
MaturityLevel ()	public:	Constructor de default
regenerateSuggestion ()	public: *void*	Recalcula la valoración sugerida debido a cambios producidos en los componentes inferires del modelo. En caso de cambiar la valoración sugerida, setea el indicador de revisión y lo propaga hacia arriba.
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

ProcessArea

public Class

Extends: AppraisalComponent. : Representa un área de proceso del modelo CMMI-SW

ProcessArea Attributes

Attribute	Type	Notes
out_of_scope	private : *boolean*	Indica si el área de proceso está fuera de alcance
SATISFIED	public const static : *String*	Constante para valoración "Satisfecha" Initial Value: "SATISFIED";
no_evidence	private : *boolean*	Indica si no existe evidencia objetiva para el área de proceso
UNSATISFIED	public const static : *String*	Constante para valoración "Insatisfecha" Initial Value: "UNSATISFIED";
NOT_APPLICABLE	public const static : *String*	Constante para valoración "No aplicable" Initial Value: "NOT_APPLICABLE";
NOT_RATED	public const static : *String*	Constante para valoración "Sin puntaje" Initial Value: "NOT_RATED";

ProcessArea Methods

Method	Type	Notes
ProcessArea (*String*)	public:	param: name [String - in] Nombre del área de proceso Constructor a partir del nombre
isOut_of_scope ()	«property get» public: *boolean*	Obtiene el valor del atributo out_of_scope
ProcessArea ()	public:	Constructor de default
setOut_of_scope (*boolean*)	«property set» public: *void*	param: newVal [boolean - in] Valor a setear Setea el valor del atributo out_of_scope
isNo_evidence ()	«property get» public: *boolean*	Obtiene el valor del atributo no_evidence
setNo_evidence (*boolean*)	«property set» public: *void*	param: newVal [boolean - in] Valor a setear Setea el valor del atributo no_evidence
generateSuggestion (*boolean, boolean*)	public: *void*	param: out_of_scope [boolean - in] Indicador de fuera de alcance param: no_evidence [boolean - in] Indicador de sin evidencia Genera una valoración sugerida, tomando en cuenta los parámetros
regenerateSuggestion ()	public: *void*	Recalcula la valoración sugerida debido a cambios producidos en los componentes inferires del modelo. En caso de cambiar la valoración sugerida, setea el indicador de revisión y lo propaga hacia arriba.
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

Goal

public Class

Extends: AppraisalComponent. : Representa un objetivo del modelo CMMI-SW.

Goal Attributes

Attribute	Type	Notes
SATISFIED	public const static : *String*	Constante para valoración "Satisfecho" Initial Value: "SATISFIED";
UNSATISFIED	public const static : *String*	Constante para valoración "Insatisfecho" Initial Value: "UNSATISFIED";

Goal Methods

Method	Type	Notes
Goal (*String*)	public:	param: name [String - in] Nombre del objetivo Constructor a partir del nombre
getSignificantWeaknesses ()	private: *int*	Obtiene la cantidad de observaciones tipo debilidad con impacto significativo asociadas a las prácticas contenidas en el objetivo
Goal ()	public:	Constructor de default
regenerateSuggestion ()	public: *void*	Recalcula la valoración sugerida debido a cambios producidos en los componentes inferires del modelo. En caso de cambiar la valoración sugerida, setea el indicador de revisión y lo propaga hacia arriba.
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

Practice

public Class

Extends: AppraisalComponent. : Representa una práctica del modelo CMMI-SW.

Practice Attributes

Attribute	Type	Notes
FULLY_IMPLEMENTED	public const static : *String*	Constante para valoración "Completamente Implementada" Initial Value: "FULLY_IMPLEMENTED";
observations	private : *List*	Observaciones vinculadas a la práctica Initial Value: new ArrayList();
direct_artifacts	private : *String*	Artefactos directos que confirman la práctica
LARGELY_IMPLEMENTED	public const static : *String*	Constante para valoración "Ampliamente Implementada" Initial Value: "LARGELY_IMPLEMENTED";
indirect_artifacts	private : *String*	Artefactos indirectos que confirman la práctica

Attribute	Type	Notes
PARTIALLY_IMPLEMENTED	public const static : *String*	Constante para valoración "Parcialmente Implementada" Initial Value: "PARTIALLY_IMPLEMENTED";
NOT_IMPLEMENTED	public const static : *String*	Constante para valoración "No Implementada" Initial Value: "NOT_IMPLEMENTED";
LARGELY_PARTIALLY_IMPLEMENTED	public const static : *String*	Constante para valoración mixta "Ampliamente o Parcialmente Implementada" Initial Value: "LARGELY_PARTIALLY_IMPLEMENTED";
NOT_PARTIALLY_LARGELY_IMPLEMENTED	public const static : *String*	Constante para valoración mixta "No Implementada, Parcialmente o Ampliamente Implementada" Initial Value: "NOT_PARTIALLY_LARGELY_IMPLEMENTED";
APPROPIATE	public const static : *String*	Constante para artefactos directos "Apropiados" Initial Value: "APPROPIATE";
NOT_APPROPIATE	public const static : *String*	Constante para artefactos directos "No apropiados" Initial Value: "NOT_APPROPIATE";
UNKNOWN	public const static : *String*	Constante para artefactos directos o indirectos "No se sabe" Initial Value: "UNKNOWN";
CONFIRM	public const static : *String*	Constante para artefactos indirectos "Confirman" Initial Value: "CONFIRM";
SUGGEST	public const static : *String*	Constante para artefactos indirectos "Sugieren" Initial Value: "SUGGEST";

Practice Methods

Method	Type	Notes
Practice (*String*)	public:	param: name [String - in] Nombre de la práctica Constructor a partir del nombre de la práctica
getDirect_artifacts ()	«property get» public: *String*	Retorna el valor de direct_artifacts
Practice ()	public:	Constructor de default
setDirect_artifacts (*String*)	«property set» public: *void*	param: newVal [String - in] Valora a setear Setea el valor de direct_artifacts
getIndirect_artifacts ()	«property get» public: *String*	Retorna el valor de indirect_artifacts
setIndirect_artifacts (*String*)	«property set» public: *void*	param: newVal [String - in] Valor a setear Setea el valor de indirect_artifacts
removeObservation (*Observation*)	public: *void*	param: obs [Observation - in] Referencia a la observación a eliminar Elimina una observación de la lista de observaciones
getObservations ()	«property get» public: *List*	Retorna la lista de observaciones
addObservation (*Observation*)	public: *void*	param: obs [Observation - in] Referencia a la observación a agregar Agrega una observación a la lista de observaciones

Method	Type	Notes
setObservations (*List*)	«property set» public: *void*	param: newVal [List - in] Lista de observaciones a setear Setea la lista de observaciones. Elimina la Práctica de las viejas observaciones y la agrega en las nuevas
generateSuggestion (*String, String, List*)	public: *void*	param: da [String - in] Valor de Artefactos directos param: ia [String - in] Valor de Artefactos indirectos param: list [List - in] Lista de observaciones Genera una valoración sugerida, tomando en cuenta los parámetros
updateObservation (*Observation*)	public: *void*	param: obs [Observation - in] Observación a actualizar Modifica una observación de la lista de observaciones
getWeaknesses (*List*)	private: *int*	param: observations [List - in] Lista de observaciones a analizar Obtiene la cantidad de observaciones tipo debilidad a partir de una lista de observaciones
regenerateSuggestion ()	public: *void*	Recalcula la valoración sugerida debido a cambios producidos en los componentes inferires del modelo. En caso de cambiar la valoración sugerida, setea el indicador de revisión y lo propaga hacia arriba.
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

Observation

public Class

Extends: AbstractModel. : Representa una observación del modelo CMMI-SW.

Observation Attributes

Attribute	Type	Notes
number	private : *Integer*	Número identificatorio de la observación
STRENGTH	public const static : *String*	Constante para representación del tipo "Fortaleza" Initial Value: "STRENGTH";
type	private : *String*	Tipo de la observación
WEAKNESS	public const static : *String*	Constante para representación del tipo "Debilidad" Initial Value: "WEAKNESS";
impact	private : *String*	Impacto de la observación
SIGNIFICANT	public const static : *String*	Constante para representación del impacto "Significativo" Initial Value: "SIGNIFICANT";

Attribute	Type	Notes
description	private : *String*	Descripción de la observación
NOT_SIGNIFICANT	public const static : *String*	Constante para representación del impacto "No significativo" Initial Value: "NOT_SIGNIFICANT";
practices	private : *List*	Lista de prácticas que tienen vinculaciones con la observación Initial Value: new ArrayList();

Observation Methods

Method	Type	Notes
Observation (*Integer, String, String, String*)	public:	param: number [Integer - in] Número identificatorio de la observación param: type [String - in] Tipo de la observación param: impact [String - in] Impacto de la observación param: description [String - in] Descripción de la observación Constructor a partir de parámetros
Observation ()	public:	Constructor de default
setNumber (*Integer*)	public: *void*	param: newVal [Integer - in] Número a setear Setea el número identificatorio de la observación
getNumber ()	«property get» public: *Integer*	Retorna el número identificatorio de la observación
setType (*String*)	«property set» public: *void*	param: newVal [String - in] Tipo a setear Setea el tipo de la observación
getType ()	«property get» public: *String*	Retorna el tipo de la observación
setImpact (*String*)	«property set» public: *void*	param: newVal [String - in] Impacto a setear Setea el impacto de la observación
getImpact ()	«property get» public: *String*	Retorna el impacto de la observación
setDescription (*String*)	«property set» public: *void*	param: newVal [String - in] Descripción a setear Setea la descripción de la observación
getDescription ()	«property get» public: *String*	Retorna la descripción de la observación

Method	Type	Notes
update (*String, String, String*)	public: *void*	param: type [String - in] Tipo a setear param: impact [String - in] Impacto a setear param: description [String - in] Descripción a setear Modifica la observación con los parámetros recibidos En caso de cambiar el tipo o el impacto, notifica el cambio a las prácticas vinculadas para regenerar las valoraciones sugeridas
notifyDelete ()	public: *void*	Notifica a los objetos Practice que tiene asociados que se ha eliminado la observación
addPractice (*Practice*)	public: *void*	param: practice [Practice - in] Práctica a agregar Agrega una práctica a la lista de prácticas que tienen vinculación con la observación
removePractice (*Practice*)	public: *void*	param: practice [Practice - in] Práctica a eliminar Elimina una práctica de la lista de prácticas que tienen vinculación con la observación
getPractices ()	public: *List*	Retorna la lista de prácticas vinculadas
setPractices (*List*)	public: *void*	param: list [List - in] Lista de prácticas vinculadas Setea la lista de prácticas vinculadas
equals (*Object*)	public: *boolean*	param: other [Object - in] Redefinición del método equals

AssessmentManager

public Class: Esta clase se encarga de la coordinación de las acciones relaciondas con la evaluación de componentes del modelo.

AssessmentManager Methods

Method	Type	Notes
AssessmentManager ()	public:	Constructor de default
generateSuggestion (*ProcessArea, boolean, boolean*)	public: *void*	param: process_area [ProcessArea - in] Referencia al objeto ProcessArea sobre el cual generar la valoración sugerida param: out_of_scope [boolean - in] Indicador de fuera de alcance param: no_evidence [boolean - in] Indicador de sin evidencia Genera una valoración sugerida en base a los parámetros de entrada (aplica a Áreas de proceso)

Method	Type	Notes
generateSuggestion (*Practice, String, String, List*)	public: *void*	param: practice [Practice - in] Referencia al objeto Practice sobre el cual generar la valoración sugerida param: da [String - in] Valor de Artefactos directos param: ia [String - in] Valor de Artefactos indirectos param: list [List - in] Lista de observaciones Genera una valoración sugerida en base a los parámetros de entrada (aplica a Prácticas)
setAssessment (*MaturityLevel, String*)	public: *void*	param: level [MaturityLevel - in] Nivel de madurez en el cual setear la valoración param: assessment [String - in] Valoración a setear Almacena todos los datos correspondientes a una valoración emitida por el usuario (aplica a Niveles de madurez)
setAssessment (*ProcessArea, boolean, boolean, String, String, String*)	public: *void*	param: process_area [ProcessArea - in] Objeto ProcessArea donde almacenar los datos param: out_of_scope [boolean - in] Indicador de fuera de alcance param: no_evidence [boolean - in] Indicador de sin evidencia param: suggestion [String - in] Valoración sugerida param: assessment [String - in] Valoración elegida param: comment [String - in] Justificación Almacena todos los datos correspondientes a una valoración emitida por el usuario (aplica a Areas de proceso)
setAssessment (*Goal, String, String, String*)	public: *void*	param: goal [Goal - in] Referencia al Goal destino de los datos param: sugg [String - in] Valoración sugerida param: assess [String - in] Valoración elegida param: comm [String - in] Justificación Almacena todos los datos correspondientes a una valoración emitida por el usuario (aplica a Objetivos)

Method	Type	Notes
setAssessment (*Practice, String, String, List, String, String, String*)	public: *void*	param: practice [Practice - in] Referencia a la práctica donde se setearán los datos param: da [String - in] Valor de Artefactos directos a setear param: ia [String - in] Valor de Artefactos indirectos a setear param: obs [List - in] Lista de observaciones a setear param: sugg [String - in] Valoración sugerida a setear param: assess [String - in] Valoración a setear param: comment [String - in] Comentario a setear Almacena todos los datos correspondientes a una valoración emitida por el usuario (aplica a Prácticas)

AppraisalManager

public Class: Esta clase se encarga de la coordinación de las acciones relaciondas con la administración de evaluaciones.

AppraisalManager Methods

Method	Type	Notes
AppraisalManager ()	public:	Constructor de default
createAppraisal (*String, String, List*)	public: *Appraisal*	param: target [String - in] Objetivo de la evaluación param: scope [String - in] Alcance de la evaluación param: instances [List - in] Instancias a evaluar Gestiona la creación de una nueva evaluación. Delega la creación de la estructura del modelo a la clase AppraisalFactory.
saveAppraisal (*Appraisal, String*)	public: *void*	param: appraisal [Appraisal - in] Evaluación a almacenar param: file [String - in] Nombre del archivo de destino Gestiona el almacenamiento de una evaluación, utilizando el subsistema persistence.
openAppraisal (*String*)	public: *Appraisal*	param: file [String - in] Nombre del archivo fuente Gestiona la recuperación de una evaluación, utilizando el subsistema persistence.

Method	Type	Notes
initLevel (*Appraisal, String*)	public: *void*	param: appraisal [Appraisal - in] Referencia al objeto Appraisal que contiene el nivel a inicializar param: name [String - in] Nombre del nivel Gestiona la inicialización de las áreas de proceso, objetivos y prácticas correspondientes a un nivel del modelo

AppraisalFactory

public Class: Esta clase se encarga de la creación de los elementos que forman parte de la estructura de una evaluación, de acuerdo al modelo CMMI-SW. Implementa una variante del patrón de diseño "*AbstractFactory*".

AppraisalFactory Attributes

Attribute	Type	Notes
scope	private : *String*	Indicador de alcance de las evaluaciones
instances	private : *List*	Instancias a evaluar
levelAreas	package : *Map*	Mapa de áreas de proceso para cada nivel Initial Value: new HashMap();
areaSGoals	package : *Map*	Mapa de objetivos específicos para cada área de proceso Initial Value: new HashMap();
goalPractices	package : *Map*	Mapa de prácticas para cada objetivo Initial Value: new HashMap();

AppraisalFactory Methods

Method	Type	Notes
AppraisalFactory (*String, List*)	public:	param: scope [String - in] Alcance de las evaluaciones param: instances [List - in] Instancias a evaluar Constructor a partir del alcance y las instancias
initMaps ()	private: *void*	Inicializa los mapas que codifican la estructura del modelo
createAppraisal (*String*)	public: *Appraisal*	param: target [String - in] Objetivo de la evaluación Crea una evaluación con los elementos de primer nivel que forman parte de su estructura (objetos MaturityLevel)
initLevel (*MaturityLevel*)	public: *void*	param: level [MaturityLevel - in] Nivel de madurez a inicializar Inicializa todos los elementos contenidos en un nivel del modelo

Method	Type	Notes
initArea (*ProcessArea*)	private: *void*	param: area [ProcessArea - in] Area de proceso a inicializar Inicializa todos los elementos contenidos en un area de proceso del modelo
initGoal (*Goal*)	private: *void*	param: goal [Goal - in] Objetivo a inicializar Inicializa todos los elementos contenidos en un objetivo del modelo
initPractice (*Practice*)	private: *void*	param: practice [Practice - in] Práctica a inicializar Inicializa todos los elementos contenidos en una práctica del modelo (instancias)
getInstances ()	public: *List*	Retorna las instancias de evaluación
getScope ()	public: *String*	Retorna el indicador de alcance
setInstances (*List*)	public: *void*	param: list [List - in] Lista de instancias de evaluación Setea las instancias de evaluación
setScope (*String*)	public: *void*	param: string [String - in] Indicadora a setear Setea el indicador de alcance
expandL2 (*MaturityLevel*)	public: *void*	param: l2 [MaturityLevel - in] Referencia al nivel de madurez L2 Expande los objetivos genéricos del nivel 2 agregándole más prácticas @param level Referencia al nivel de madurez L2

ObservationManager

public Class: Esta clase se encarga de la coordinación de las acciones relacionadas con la administración de observaciones.

ObservationManager Methods

Method	Type	Notes
ObservationManager ()	public:	Constructor de default
getNumber (*List*)	private: *Integer*	param: list [List - in] Lista de observaciones existentes (objetos tipo Observation) Obtiene un número identificatorio de observación en base a las observaciones existentes.

Method	Type	Notes
createObservation (*Appraisal, String, String, String*)	public: *void*	param: appraisal [Appraisal - in] Objeto Appraisal donde ubicar la observación param: type [String - in] Tipo de la observación param: impact [String - in] Impacto de la observación param: description [String - in] Descripción de la observación Crea una nueva observación y la agrega a la evaluación recibida como argumento
updateObservation (*Appraisal, Integer, String, String, String*)	public: *void*	param: appraisal [Appraisal - in] Evaluación en la cual se debe modificar la observación param: number [Integer - in] Número de la observación a modificar param: type [String - in] Tipo a setear en la observación param: impact [String - in] Impacto a setear en la observación param: description [String - in] Descripción a setear en la observación Gestiona la modificación de una observación existente
deleteObservation (*Appraisal, Integer*)	public: *void*	param: appraisal [Appraisal - in] Evaluación sobre la que se debe eliminar la observación param: number [Integer - in] Número identificatorio de la observación a eliminar Gestiona la eliminación de una observación existente

SystemException

public Class

Implements: Exception. : Clase que encapsula todas las excepciones relacionadas con fallos en el sistema que no dependen de la lógica de negocios de la aplicación. Se utiliza para notificar excepciones a los controladores y las vistas.

SystemException Methods

Method	Type	Notes
SystemException (*String*)	public:	param: message [String - in] Mensaje asociado a la excepción Constructor a partir de un mensaje. Delega al constructor de la superclase.
SystemException (*String, Throwable*)	public:	param: message [String - in] Mensaje asociado a la excepción param: cause [Throwable - in] Excepción original Constructor a partir de un mensaje y una excepción.

7.6 Diseño de la realización de los casos de uso

El siguiente reporte de *Enterprise Architect* muestra el funcionamiento del mecanismo de inicialización del sistema. El mismo se explica mediante diagramas de secuencia.

Reporte: Inicialización del sistema

En el diagrama de la figura 7.23 se muestran las interacciones que se producen durante el arranque del sistema.

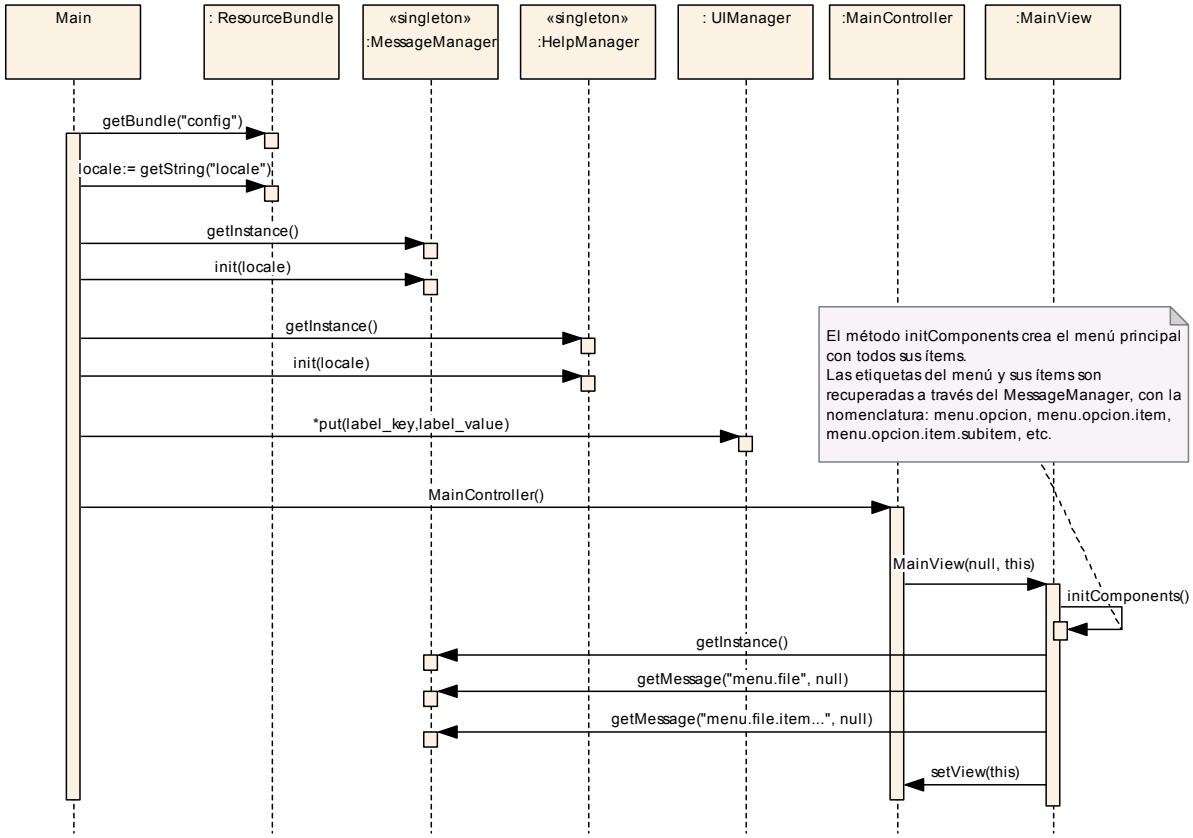

Figura 7.23. Inicialización del sistema. El programa principal (clase *Main*) obtiene un *ResourceBundle* que mapea al archivo de configuración de la aplicación (*config.properties*). A continuación le solicita al *ResourceBundle* la variable de configuración "*locale*" que indica el idioma a utilizar en la aplicación. Con el valor obtenido, inicializa el subsistema de mensajes mediante el método *init* del *MessageManager* y el subsistema de ayudas y guías online mediante el método *init* del *HelpManager*.
A continuación, define las las etiquetas correspondientes a las ventanas provistas por la plataforma J2SE (ventanas de confirmación, de mensajes de error, y de apertura y almacenamiento de archivos) invocando al método *put* del *UIManager* (objeto provisto por la plataforma J2SE)
Finalmente, instancia un objeto de la clase *MainController*, quien se encarga de instanciar a su vez la ventana principal de la aplicación (*MainView*). Durante su inicialización (método *initComponents*), la *MainView* accede a la única instancia del *MessageManager* mediante el método *getInstance*, y obtiene

todas las etiquetas que forman parte del menú principal de la aplicación. Con las etiquetas obtenidas construye el menú principal de la aplicación. Por último, la *MainView* se registra como vista del *MainController* mediante el mensaje *setView*.

El siguiente reporte muestra el funcionamiento del mecanismo de ayudas y guías online. El mismo se explica mediante diagramas de secuencia.

Reporte: Utilización de las ayudas y guías

En el diagrama de la figura 7.24 se muestran las interacciones que se producen cuando cualquier vista del sistema desea utilizar guías o ayudas.

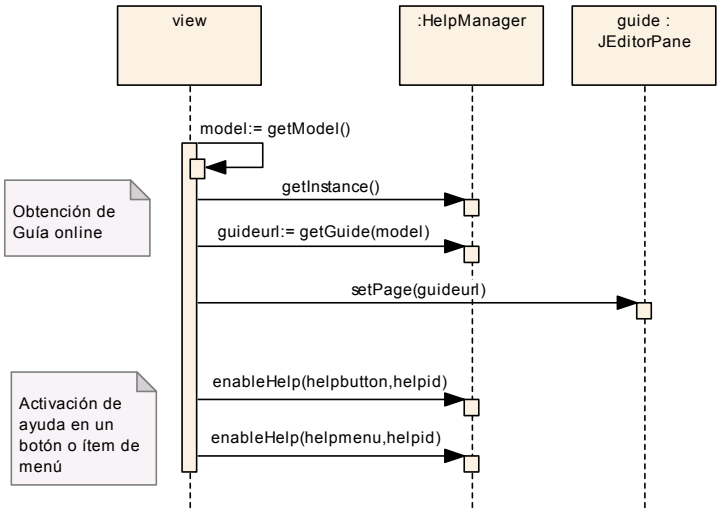

Figura 7.24. Utilización de las ayudas y guías. Para la obtención de la guía online asociada a un componente, la vista obtiene el componente (Nivel de madurez, Área de proceso, Objetivo o Práctica) mediante el método *getModel*, instancia al *HelpManager* (*getInstance*), le solicita la guía (*getGuide*) pasándole el componente como argumento, y finalmente setea la url obtenida en un componente del tipo *JEditorPane* (panel que muestra archivos html). Para la activación de las ayudas, la vista invoca al método *enableHelp* del *HelpManager* pasándole como argumentos el botón o el ítem de menú a activar, y el identificador de la ayuda a mostrar para el mismo.

El siguiente reporte muestra las realizaciones de caso de uso de diseño, obtenidas como resultado de la actividad "Diseño de casos de uso reales" contemplada en la metodología.

Reporte: Diseño de la realización de los casos de uso

El diseño de la realización de los casos de uso muestra la forma en que interactúan las clases de diseño dentro del sistema para llevar a cabo la funcionalidad descripta en los casos de uso.

Para cada caso de uso del sistema, se efectúa una "realización" de diseño donde se documentan las clases que participan en el mismo y la manera en que interactúan entre sí para llevar a cabo la funcionalidad especificada. Las interacciones entre clases se documentan mediante "Diagramas de secuencia".

Cada realización contiene uno o más diagramas de clase, donde se muestran las clases involucradas en la realización, y uno o más diagramas de secuencia, donde se muestran las interacciones entre los objetos de esas clases.

El diagrama de la figura 7.25 muestra gráficamente las relaciones de trazabilidad existentes entre las realizaciones de diseño y los casos de uso.

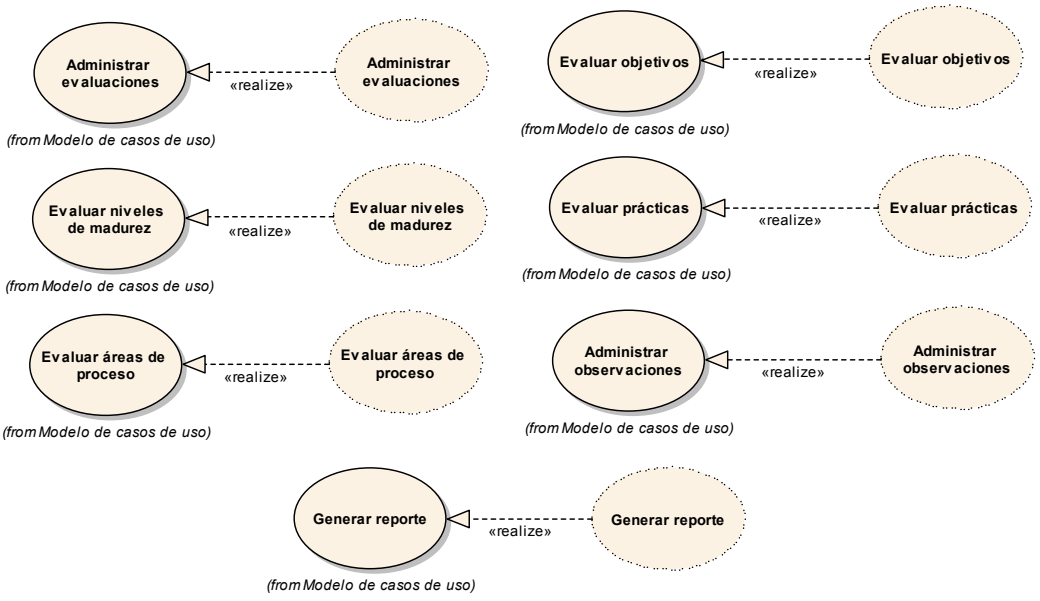

Figura 7.25. Trazabilidad.

A continuación se detallan una a una todas las realizaciones de diseño.

Administrar evaluaciones

El diagrama de la figura 7.26 muestra las clases que participan en la realización.

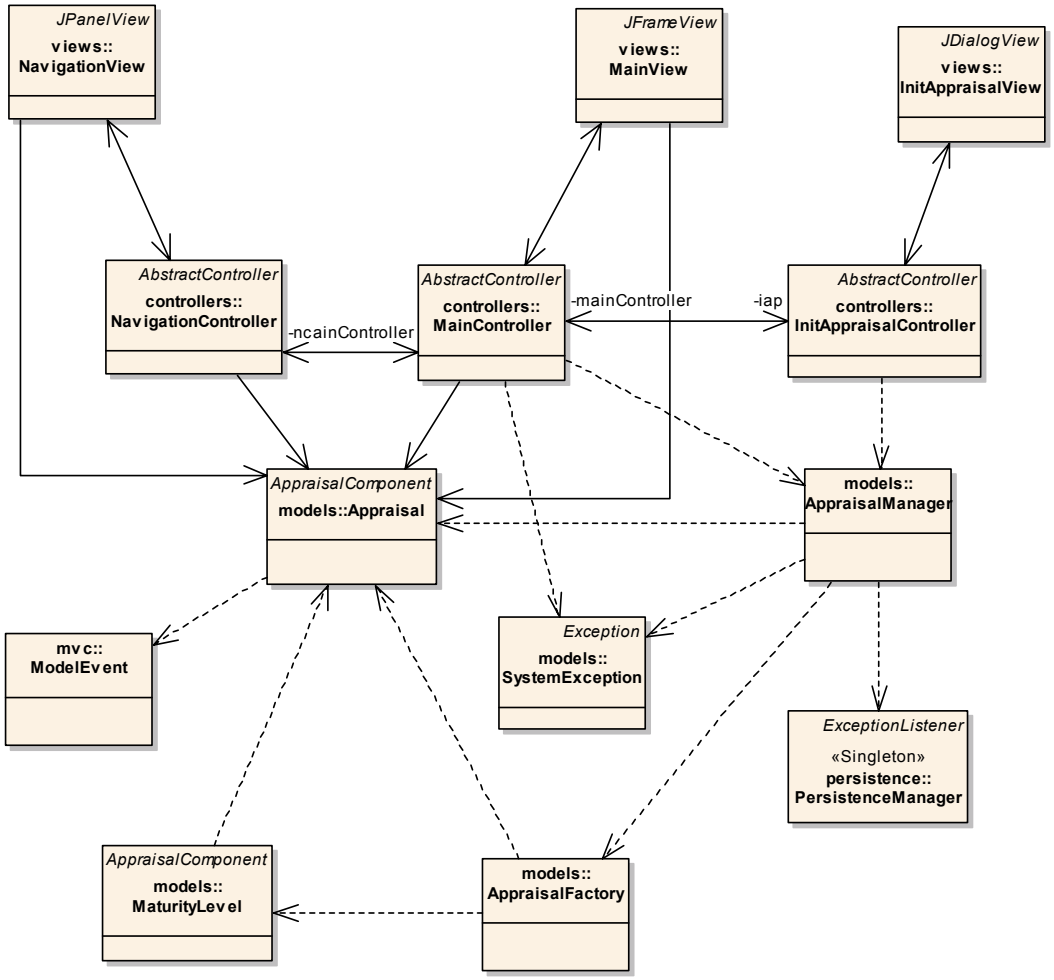

Figura 7.26. Participantes.

Los diagramas de las figuras 7.27a, 7.27b y 7.27c muestran las interacciones que se producen en el inicio de una evaluación.

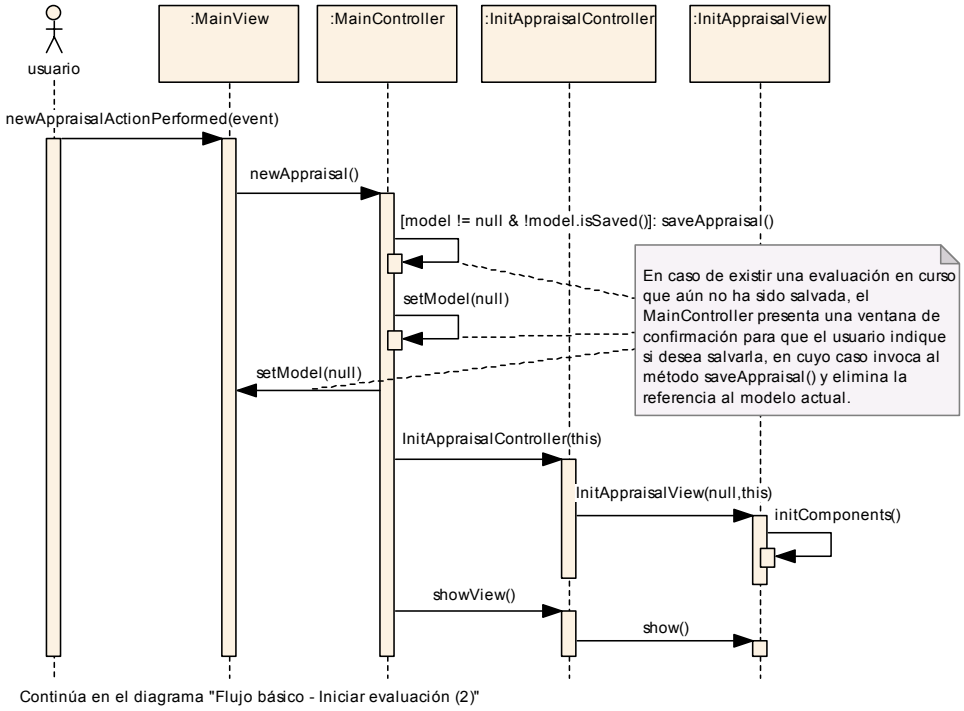

Figura 7.27a. Flujo básico - Iniciar evaluación (1). El usuario selecciona la opción de menú "Nueva evaluación" en la *MainView*, lo que se traduce en la invocación del método *newAppraisalActionPerformed* en la misma. La *MainView* invoca el método *newAppraisal* del *MainController*.El *MainController* verifica si existe una evaluación en curso y si necesita ser almacenada, en cuyo caso presenta una *JOptionPane* al usuario para que confirme si desea almacenarla. Si el usuario confirma invoca al método *saveAppraisal*() desencadenando las interacciones del diagrama "7.28. Flujo alternativo - Almacenar evaluación". La evaluación es almacenada y el *MainController* elimina su modelo actual y lo elimina de la *MainView*. Luego de eso, el *MainController* crea una instancia del *InitAppraisalController* pasándole una referencia a él mismo como argumento. El *InitAppraisalController* crea una instancia de la *InitAppraisalView*, a la cual le pasa como argumento una referencia a sí mismo. La *InitAppraisalView* inicializa la interfaz de usuario (*initComponents*). Finalmente, el *MainController* invoca al método *showView* del *InitAppraisalController* para que muestre su vista asociada.

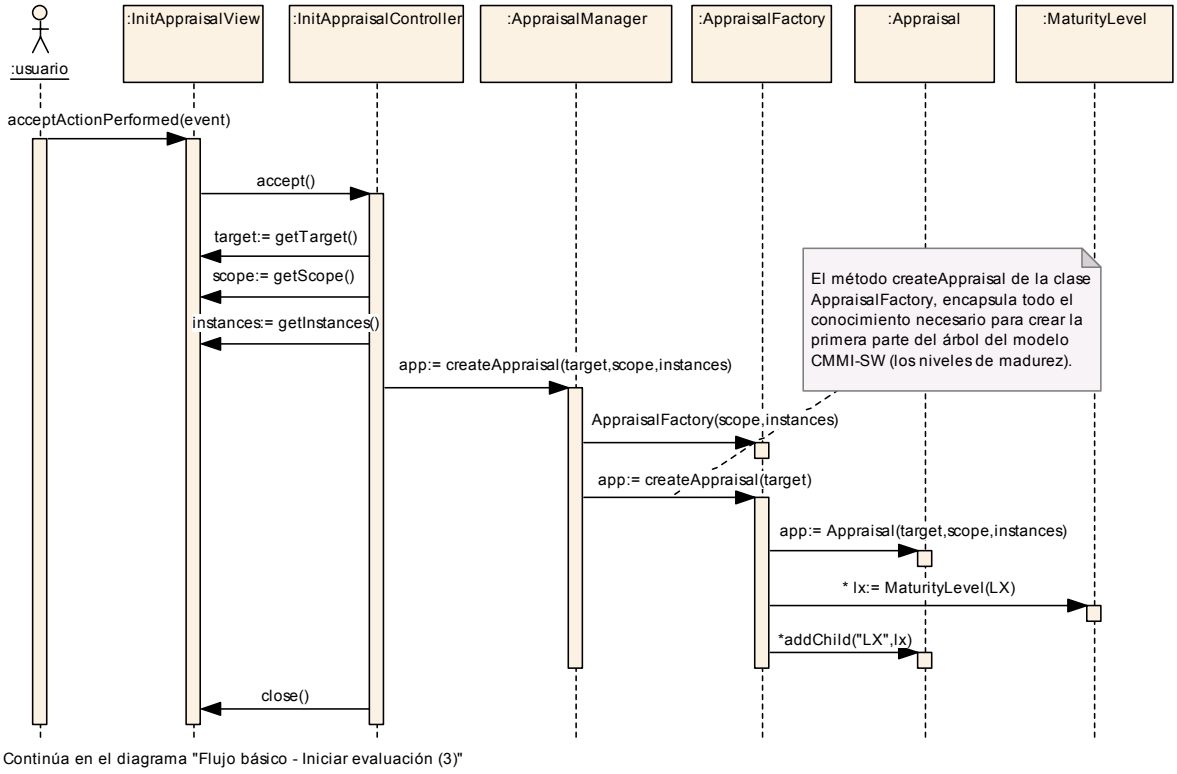

Figura 7.27b. Flujo básico - Iniciar evaluación (2). El usuario ingresa los datos de la evaluación a iniciar y presiona el botón Aceptar en la ventana, lo que se traduce en la invocación del método *acceptActionPerformed* en la *InitAppraisalView*. La vista invoca al método *accept* de su controlador asociado (*InitAppraisalController*) para procesar la acción del usuario. El *InitAppraisalController* recupera los datos de la interfaz (*getTarget*, *getScope*, *getInstances*) e invoca al método *createAppraisal* del *AppraisalManager*, quien delega la creación en el *AppraisalFactory*. El *AppraisalFactory* crea todos los objetos de primer nivel (*MaturityLevel*) y los anida de acuerdo a la estructura del árbol que conforma el modelo CMMI-SW. Finalmente, el *InitAppraisalController* cierra la interfaz *InitAppraisalView*.

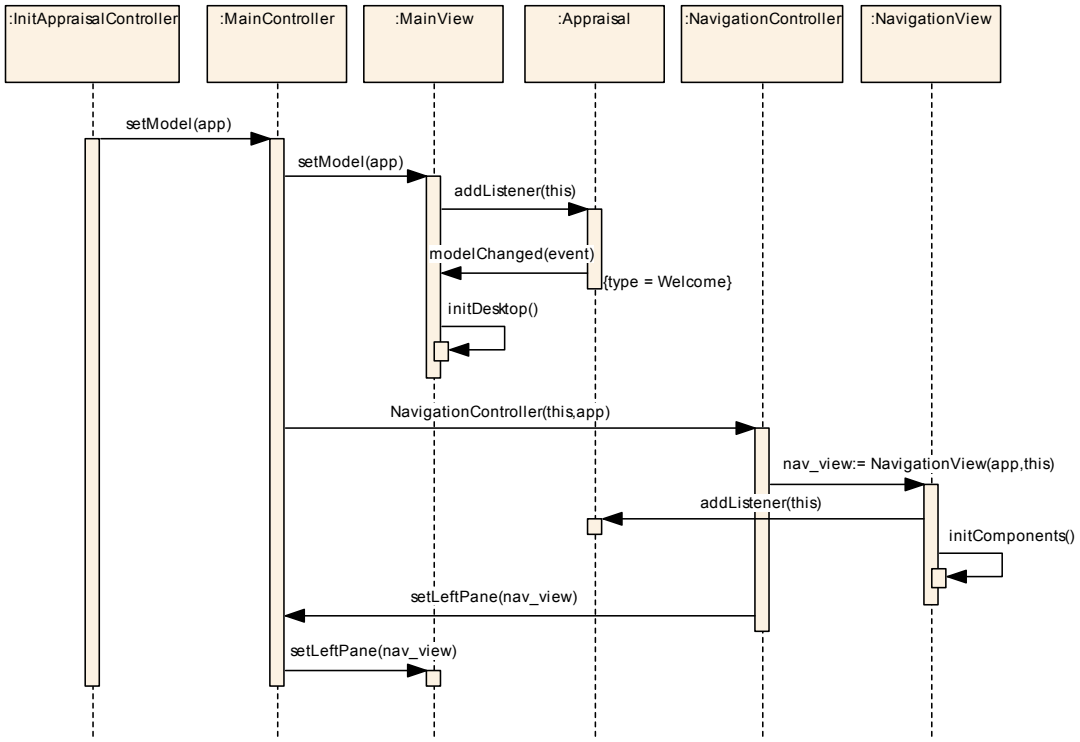

Figura 7.27c. Flujo básico - Iniciar evaluación (3). El *InitAppraisalController* entrega el objeto *Appraisal* que acaba de crearse al *MainController* (*setModel*), quien a su vez lo propaga a la *MainView* (*setModel*). La *MainView* se registra como *listener* del modelo y recibe la notificación de bienvenida (*Welcome*) ante lo cual inicializa el desktop (*initDesktop*). El *MainController* instancia al *NavigationController* pasandole una referencia a sí mismo y otra al *Appraisal* que acaba de crearse, de manera que lo utilice como modelo. El *NavigationController* crea la *NavigationView* pasandole el *Appraisal* como modelo. La *NavigationView* inicializa la interfaz de usuario (*initComponents*) y se registra como listener del modelo (*Appraisal*). La interfaz de navegación es un árbol JTree en el que se despliegan todos los elementos de primer nivel (*AppraisalComponent*) que forman parte del *Appraisal* recibido como modelo. Para construir el JTree, se invoca el método *getChildren* de la clase *AppraisalComponent*, obteniendo la lista de niveles de madurez. Finalmente, el *NavigationController* entrega la *NavigationView* al *MainController* para que la ubique en el panel izquierdo de la *MainView* (*setLeftPane*).

El diagrama de la figura 7.28 muestra las interacciones que se producen en el almacenamiento de una evaluación.

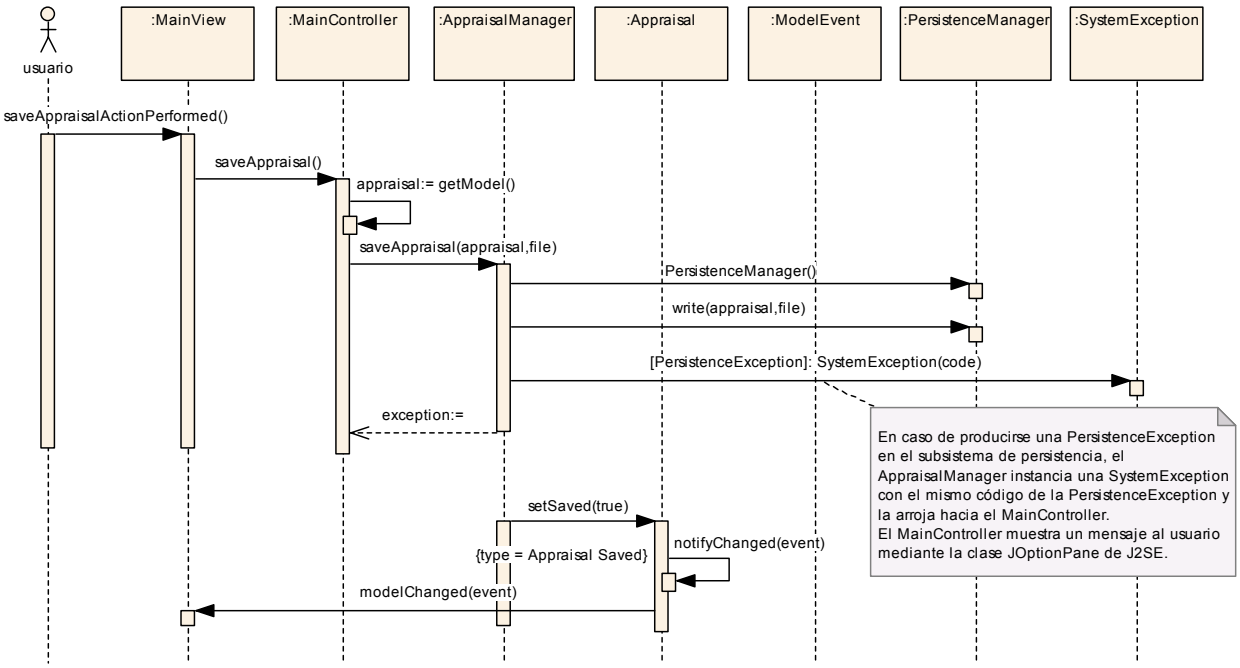

Figura 7.28. Flujo alternativo - Almacenar evaluación. El usuario selecciona la opción "Guardar" del menú principal, lo que se traduce en la invocación del método *saveAppraisalActionPerformed* de la *MainView*. La *MainView* invoca al método *saveAppraisal* del *MainController*.El *MainController* presenta al usuario una ventana estándar para la selección de archivos utilizando la clase *JFileChooser* de J2SE en la cual el usuario indica un nombre de archivo (*file*). Una vez que el usuario indicó el archivo, el *MainController* rescata su modelo asociado (*Appraisal*), instancia al *AppraisalManager* y le envía el mensaje *saveAppraisal*, pasándole como parámetros el modelo y el nombre de archivo indicado por el usuario (*appraisal* y *file*).El *AppraisalManager* instancia al subsistema de persistencia (*PersistenceManager*) e invoca al método *write* del mismo, pasándole la evaluación a almacenar y el nombre de archivo de destino. En caso de excepciones, el subsistema de persistencia arroja una *PersistenceException* con un código indicativo del error. El *AppraisalManager* instancia una *SystemExceptión* con el mismo código y la arroja hacia el *MainController*.En caso de no haber excepciones, el *AppraisalManager* invoca el método *setSaved* con parámetro true para notificar al *Appraisal* que ha sido almacenado correctamente. El *Appraisal* notifica a los interesados (*listeners*) mediante un *ModelEvent* de tipo "*Appraisal Saved*". Entre los interesados figura la *MainView*, quien inhabilita la opción Guardar de su menú al recibir la notificación.

El diagrama de la figura 7.29 muestra las interacciones que se producen en la recuperación de una evaluación.

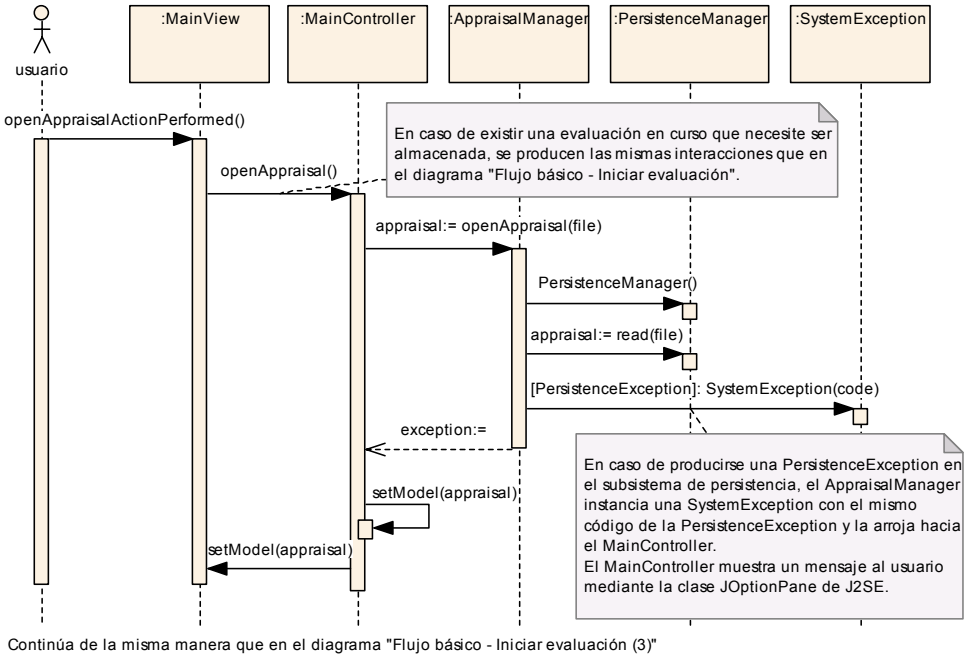

Figura 7.29. Flujo alternativo - Recuperar evaluación. El usuario selecciona la opción "Abrir" del menú principal, lo que se traduce en la invocación del método *openAppraisalActionPerformed* de la *MainView*. La *MainView* invoca al método *openAppraisal* del *MainController*. El *MainController* presenta al usuario una ventana estándar para la selección de archivos utilizando la clase *JFileChooser* de J2SE en la cual el usuario indica un nombre de archivo (*file*). Una vez que el usuario indicó el archivo, el *MainController* instancia al *AppraisalManager* y le envía el mensaje *openAppraisal*, pasándole como parámetro el nombre de archivo indicado por el usuario (*file*). El *AppraisalManager* instancia al subsistema de persistencia (*PersistenceManager*) e invoca al método *read* del mismo, pasándole el nombre de archivo fuente y obteniendo como resultado la evaluación almacenada (objeto *Appraisal*). En caso de excepciones, el subsistema de persistencia arroja una *PersistenceException* con un código indicativo del error. El *AppraisalManager* instancia una *SystemExceptión* con el mismo código y la arroja hacia el *MainController*, quien se encarga de mostrar el error al usuario mediante una *JOptionPane* (previa conversión del código de error a mensaje mediante el subsistema de mensajes). En caso de no haber excepciones, el *MainController* setea como modelo la evaluación recuperada, y la propaga a la *MainView*. De allí en adelante, se producen las mismas interacciones que en el diagrama "7.27c. Flujo básico - Iniciar evaluación (3)."

Evaluar niveles de madurez

El diagrama de la figura 7.30 muestra las clases que participan en la realización.

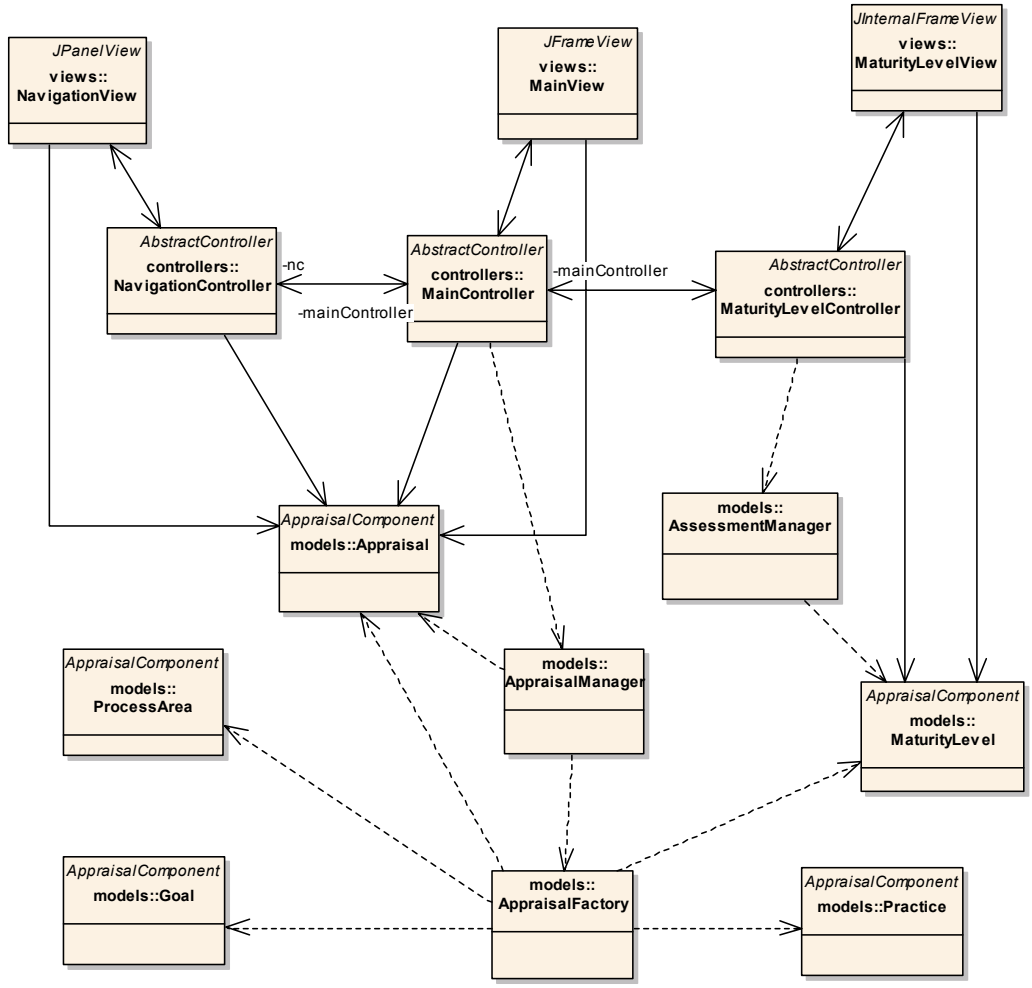

Figura 7.30. Participantes.

Los diagramas de las figuras 7.31a, 7.31b, 7.31c, 7.31d y 7.31e muestran las interacciones que se producen en la evaluación de un nivel de madurez.

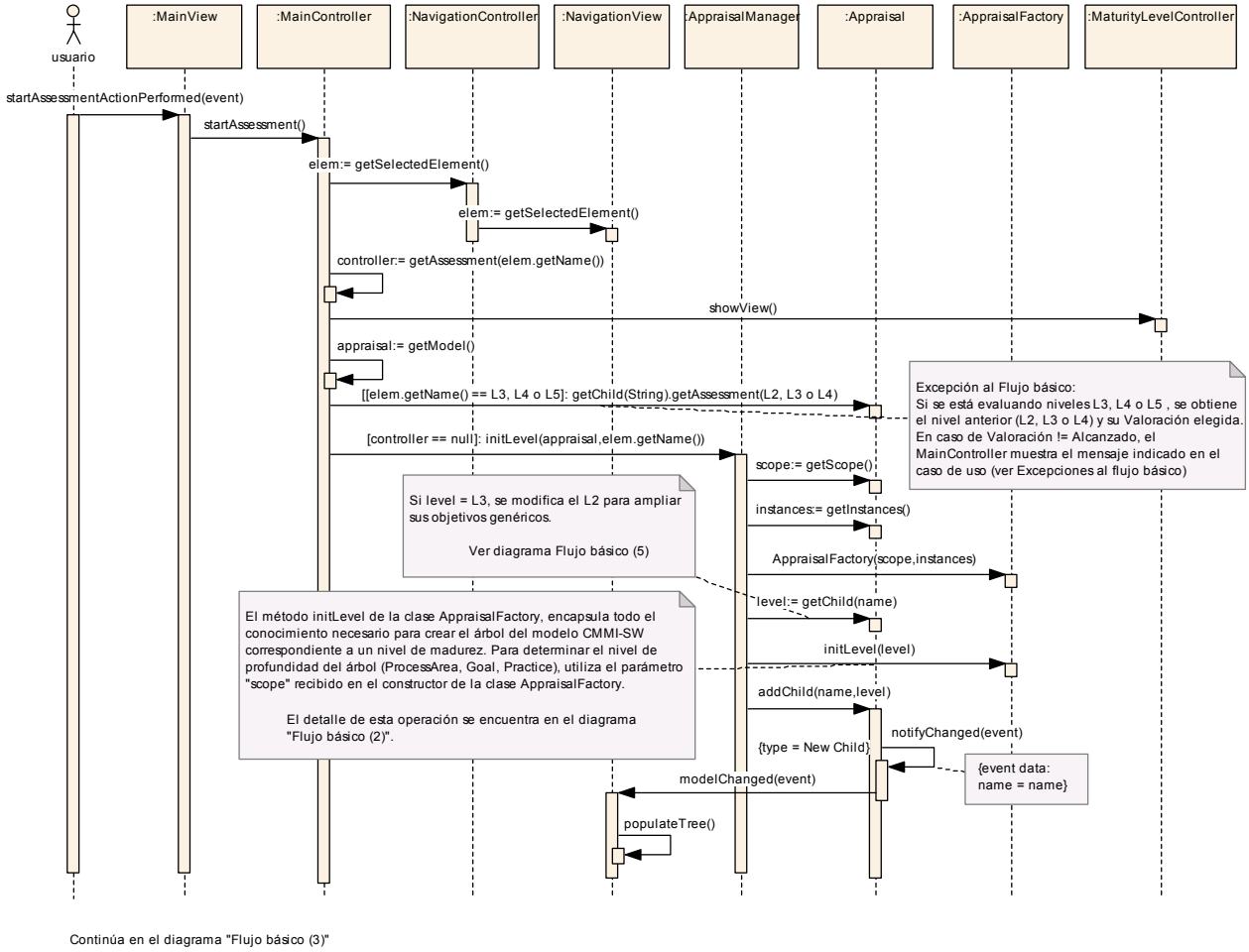

Figura 7.31a. Flujo básico (1). El usuario selecciona un nivel de madurez en el árbol de navegación del modelo y a continuación selecciona la opción del menú "Evaluar". Esto se traduce en la invocación del método *startAssessmentActionPerformed* en la *MainView*. La *MainView* invoca al método *startAssessment* del *MainController*. El *MainController* obtiene el elemento seleccionado en el árbol mediante el método *getSelectedElement* del *NavigationController* (el cual a su vez utiliza el *getSelectedElement* de la *NavigationView*). Con el elemento obtenido, verifica si ya existe una evaluación en curso para ese elemento, en cuyo caso ordena al controlador asociado a la misma que presente la interfaz al usuario y termina la secuencia. En caso de no existir una evaluación en curso, el *MainController* instancia al *AppraisalManager* y le envía el mensaje *initLevel*, pasándole como argumentos el objeto *Appraisal* que tiene como modelo y el nombre del elemento seleccionado en el árbol. El *AppraisalManager* recupera el alcance (*getScope*) y las instancias de evaluación (*getInstances*) e inicializa un objeto *AppraisalFactory* para delegarle la inicialización del nivel de madurez seleccionado (método *initLevel* de *AppraisalFactory*). Luego de inicializar el nivel, el *AppraisalManager* agrega nuevamente el nivel a la evaluación (método *addChild*), lo que desencadena que la evaluación (objeto *Appraisal*) genere un *ModelEvent* para notificar a todos los interesados en sus cambios, entre los cuales se encuentra la *NavigationView*. Esta vista se encarga de completar los nodos y hojas del JTree para reflejar los cambios en el modelo cuando recibe la notificación del cambio.

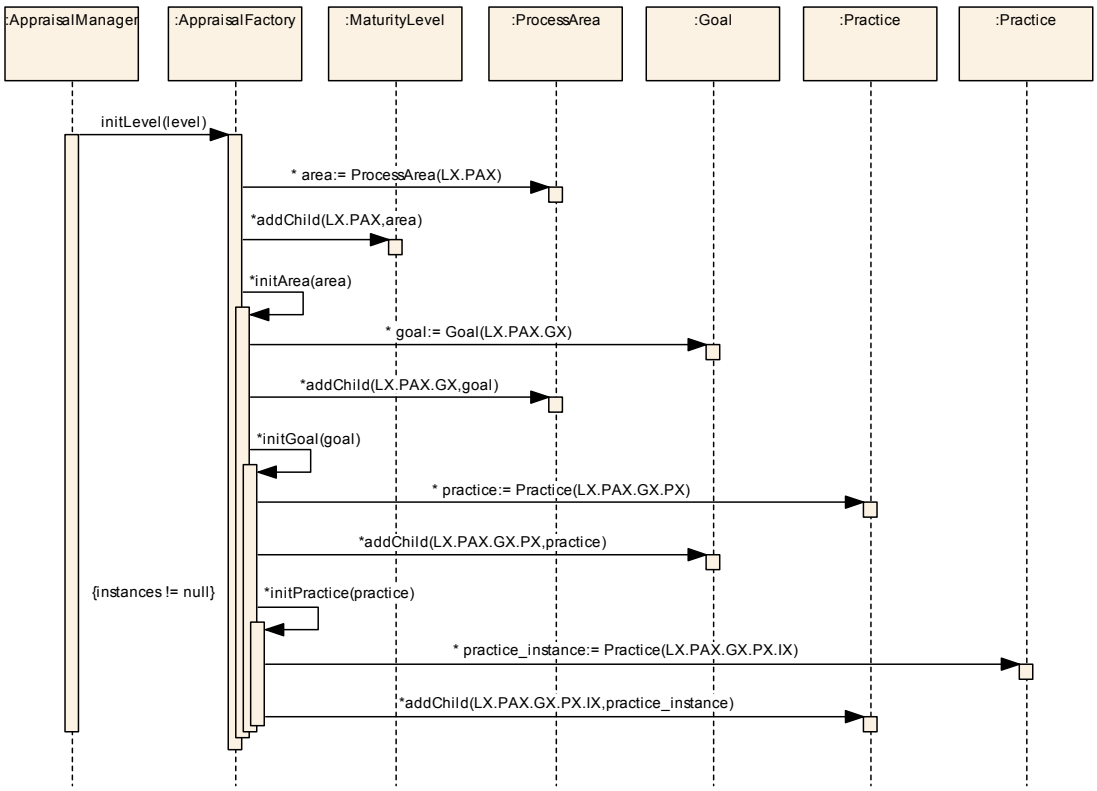

Figura 7.31b. Flujo básico (2). El *AppraisalManager* invoca el método *initLevel* del *AppraisalFactory*, pasándole como argumento el nivel a inicializar (objeto *MaturityLevel*). Para cada área de proceso del modelo, el *AppraisalFactory* instancia un objeto *ProcessArea*, lo agrega al nivel recibido como parámetro, e invoca al método *initArea* para inicializar el área creada. El método *initArea* crea los objetivos del área de proceso, los agrega a la misma, e invoca al método *initGoal* para inicializar cada uno de los objetivos. El método *initGoal* crea las prácticas del objetivo y las agrega al mismo. Para cada práctica creada, si existen instancias de evaluación se crea una práctica para cada instancia y se incluye dentro de la práctica madre, con lo cual se tienen dos niveles de prácticas (el de instancia y el de conjunto de instancias).

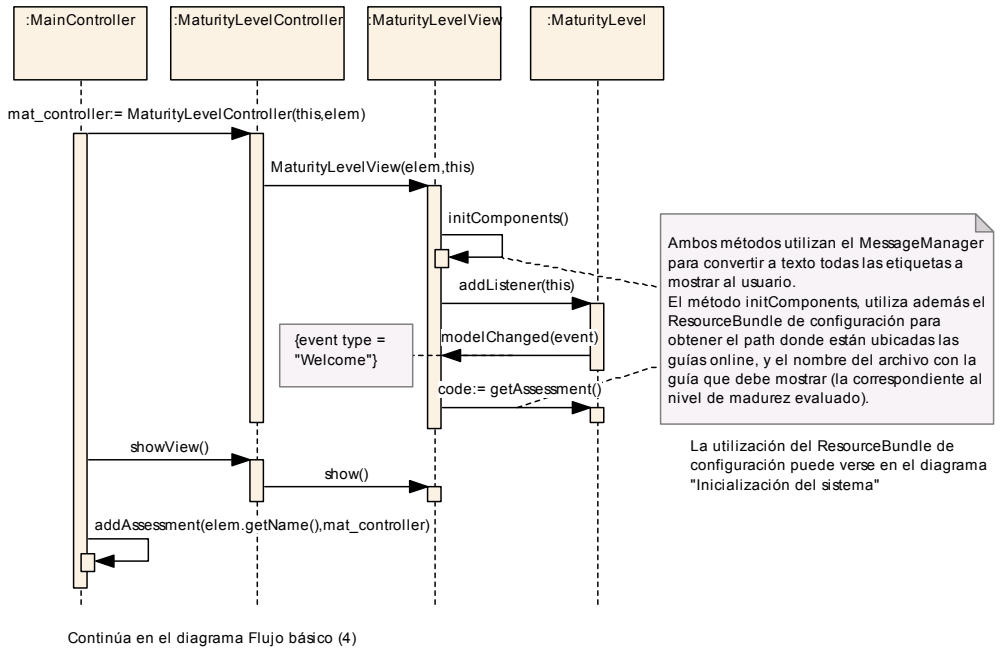

Figura 7.31c. Flujo básico (3). El *MainController* instancia un *MaturityLevelController* pasándole como argumentos una referencia a sí mismo y una referencia al *MaturityLevel* que tiene como modelo (objeto *elem* de tipo *AppraisalComponent*). El *MaturityLevelController* instancia una *MaturityLevelView* pasándole una referencia al modelo (*MaturityLevel*) y otra a sí mismo. La *MaturityLevelView* inicializa la interfaz de usuario (*initComponents*), y se registra como *listener* de su modelo (*MaturityLevelView*), quien le envía como respuesta una notificación de bienvenida (*modelChanged* con evento "*Welcome*"). Al recibir la notificación, la *MaturityLevelView* obtiene la puntuación asociada al nivel mediante el método *getAssessment* y la vuelca en el campo "Valoración concluída" de la interfaz de usuario. Finalmente, el *MainController* envía el mensaje *showView* al *MaturityLevelController* y lo agrega al pool de controladores de evaluación, para registrarlo como evaluación en curso.

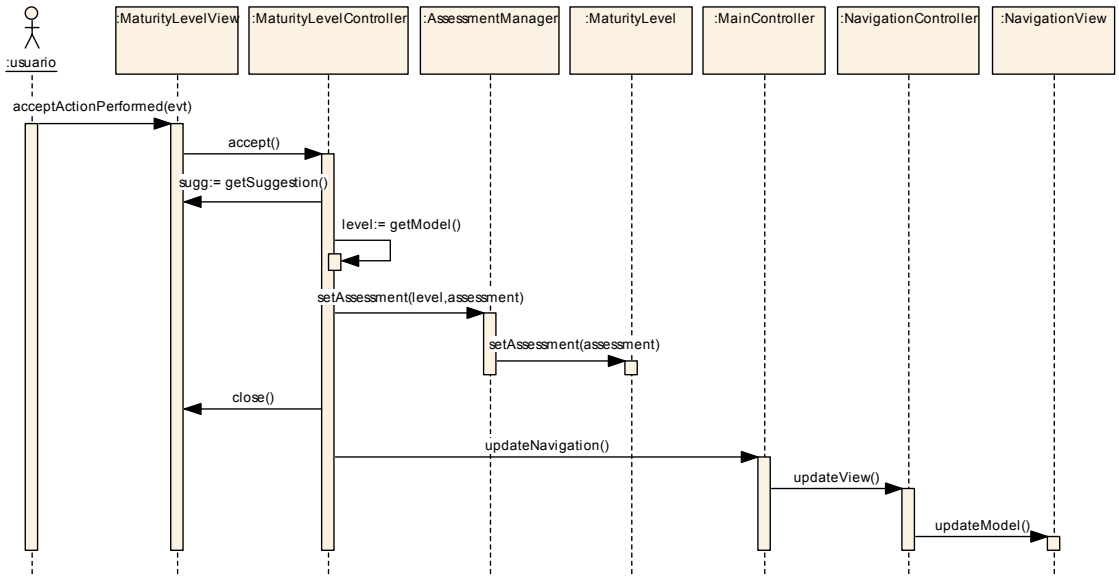

Figura 7.31d. Flujo básico (4). El usuario presiona el botón aceptar, lo que se traduce en la invocación del método *acceptActionPerformed* en la *MaturityLevelView*. La vista delega el procesamiento al método

accept del *MaturityLevelController*, quien recupera la valoración sugerida de la interfaz, su modelo asociado (*level*), e invoca al método *setAssessment* del *AssessmentManager*, para setear la valoración en el objeto *MaturityLevel*. Luego de eso, el *MaturityLevelController* cierra la vista e invoca al método *updateNavigation* del *MainController*, quien a su vez invoca al *updateView* del *NavigationController* para refrescar el árbol del modelo (*updateModel*) en la *NavigationView*.

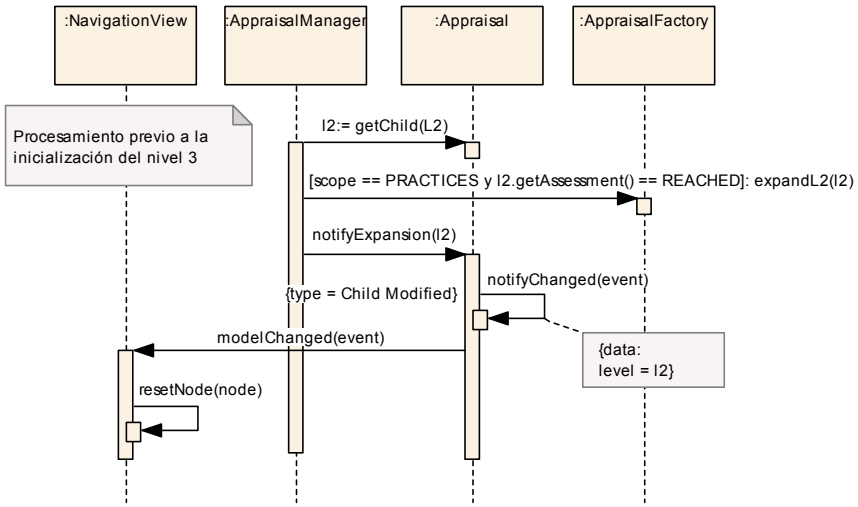

Figura 7.31e. Flujo básico (5). Como paso previo a la inicialización del nivel 3, el *AppraisalManager* obtiene el nivel 2 del *Appraisal* (*getChild*). Si el alcance de la evaluación es a nivel de prácticas y el nivel 2 se encuentra Alcanzado, el *AppraisalManager* invoca al método *expandL2* del *AppraisalFactory*. El método *expandL2* se encarga de ampliar los objetivos genéricos del nivel 2 agregándole prácticas. Luego de expandir el nivel 2, el *AppraisalManager* invoca al método *notifyExpansion* de *Appraisal*, quien se encarga de componer un *ModelEvent* y notificar a los interesados que se ha producido la expansión del nivel 2 (evento tipo "*Child Modified*" con el *AppraisalComponent* que representa al nivel 2 como parámetro asociado). Finalmente, la *NavigationView* recibe el evento mediante el método *modelChanged* y resetea el nodo modificado (*resetNode*).

Evaluar áreas de proceso

El diagrama de la figura 7.32 muestra las clases que participan en la realización.

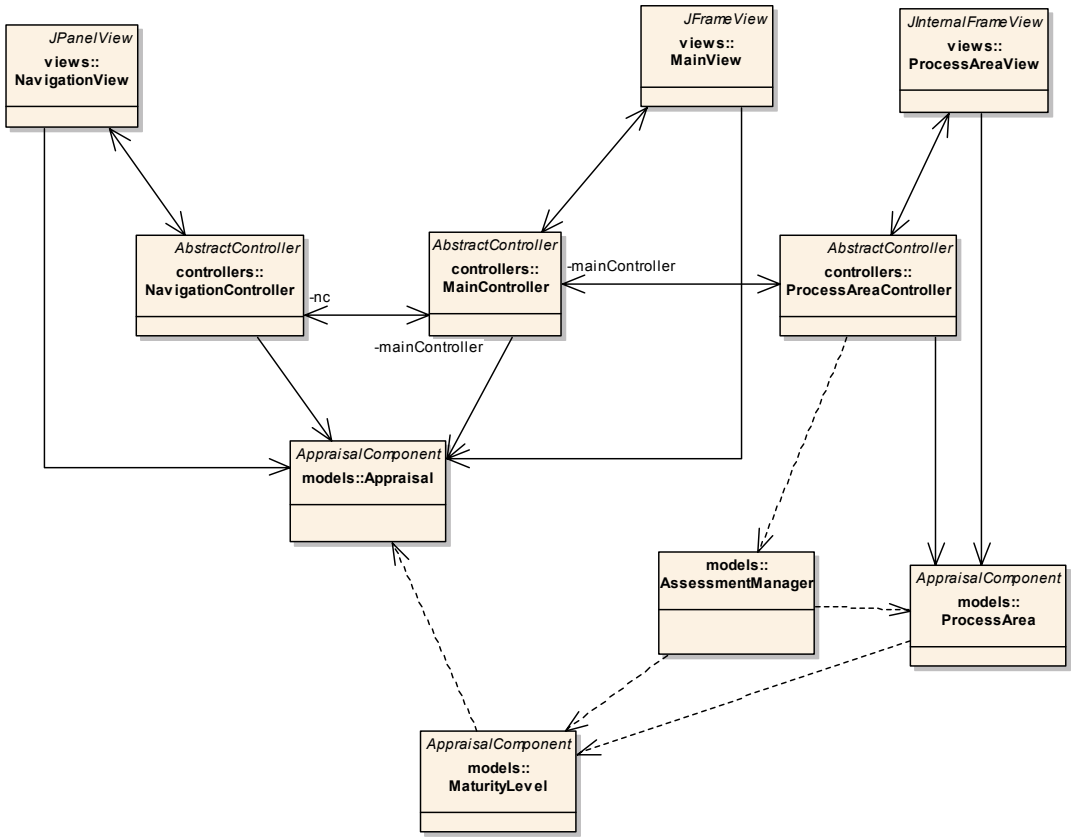

Figura 7.32. Participantes.

Los diagramas de las figuras 7.33a, 7.33b, 7.33c y 7.33d muestran las interacciones que se producen en la evaluación de un área de proceso.

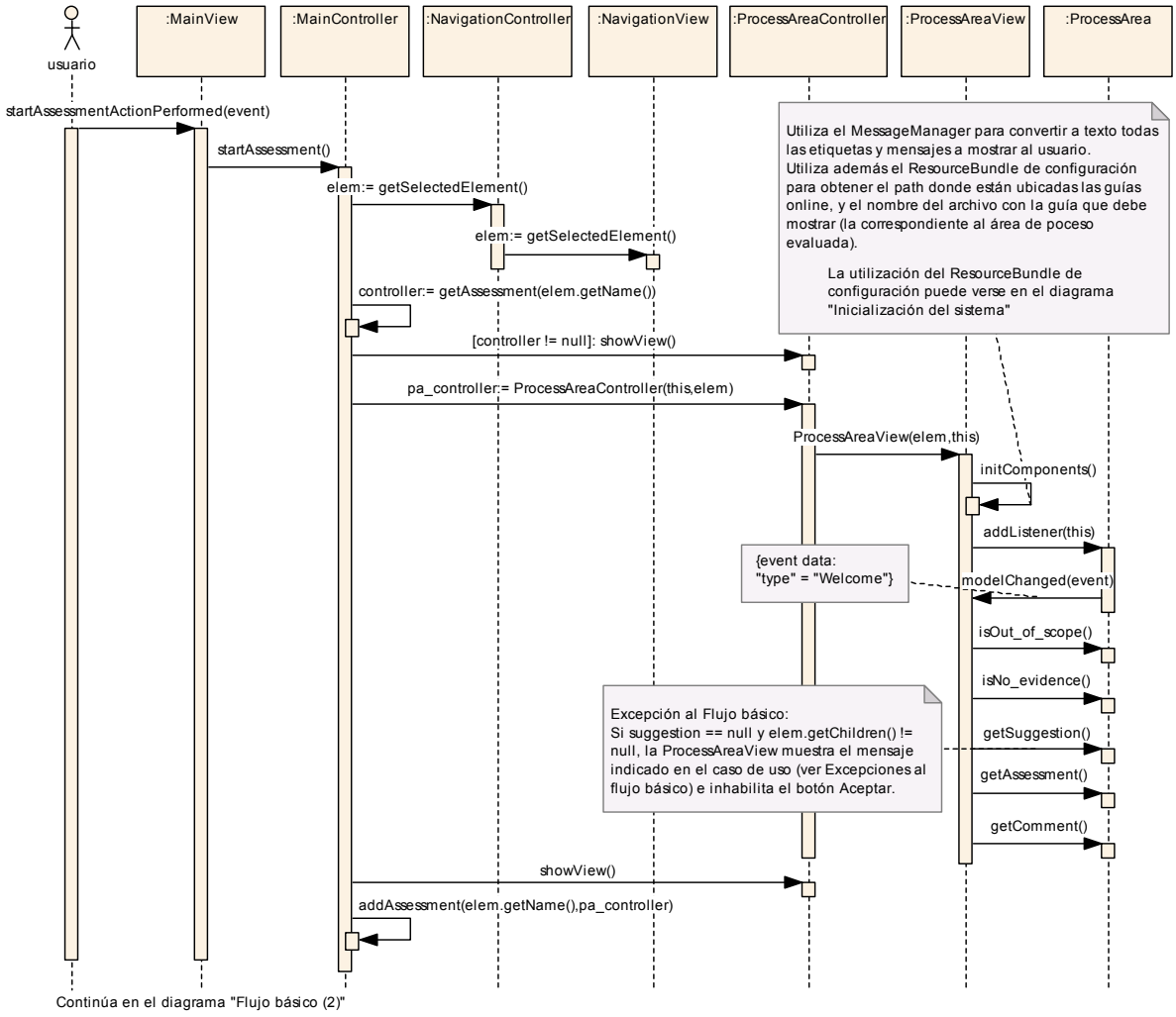

Figura 7.33a. Flujo básico (1). El usuario selecciona un área de proceso en el árbol de navegación del modelo y a continuación selecciona la opción del menú "Evaluar". Esto se traduce en la invocación del método *startAssessmentActionPerformed* en la *MainView*. La *MainView* invoca al método *startAssessment* del *MainController*. El *MainController* obtiene el elemento seleccionado en el árbol mediante el método *getSelectedElement* del *NavigationController* (el cual a su vez utiliza el *getSelectedElement* de la *NavigationView*). Con el elemento obtenido, verifica si ya existe una evaluación en curso para ese elemento, en cuyo caso ordena al controlador asociado a la misma que presente la interfaz al usuario y termina la secuencia. En caso de no existir una evaluación en curso, el *MainController* instancia un *ProcessAreaController* pasándole como argumentos una referencia a sí mismo y una referencia al *ProcessArea* que tiene como modelo (objeto *elem* de tipo *AppraisalComponent*). El *ProcessAreaController* instancia una *ProcessAreaView* pasándole una referencia al modelo (*ProcessArea*) y otra a sí mismo. La *ProcessAreaView* inicializa la interfaz de usuario (*initComponents*), y se registra como listener de su modelo, quien le envía como respuesta una notificación de bienvenida (*modelChanged* con evento "*Welcome*"). Al recibir la notificación, la *ProcessAreaView* obtiene los atributos del área de proceso (métodos *isOut_of_scope*, *isNo_evidence*, *getSuggestion*, *getAssessment*, *getComment*) y la vuelca en los campos de la interfaz de usuario. Si no existe una valoración sugerida y el área de proceso tiene hijos, la *ProcessAreaView* muestra un mensaje de advertencia al usuario (mensaje explicitado en el caso de uso, Excepción al flujo básico). Finalmente, el *MainController* indica al *ProcessAreaController* que muestre su vista asociada (*showView*) y lo agrega al pool de controladores de evaluación, para registrarlo como evaluación en curso.

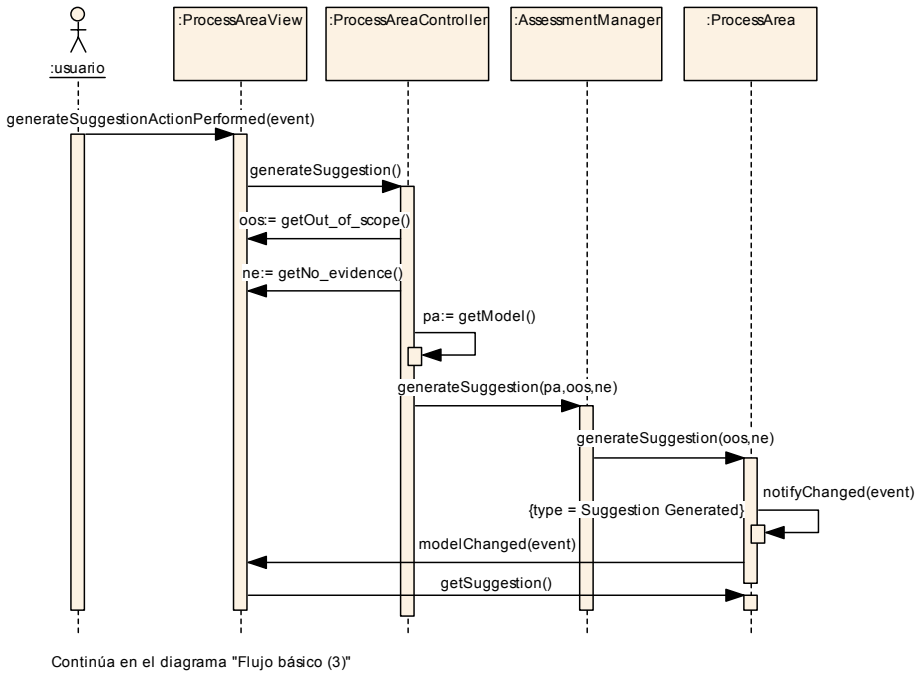

Figura 7.33b. Flujo básico (2). El usuario modifica los campos "Fuera de alcance" o "Sin evidencia" en la interfaz de evaluación, lo que se traduce en la invocación del método *generateSuggestionActionPerformed* en la *ProcessAreaView*. La *ProcessAreaView* invoca el método *generateSuggestion* del *ProcessAreaController*, quien se encarga de recuperar los valores de la ventana y obtener su modelo asociado (objeto *ProcessArea*). A continuación, instancia al *AssessmentManager* e invoca al método *generateSuggestion* pasándole como argumentos la *ProcessArea* y los valores de la ventana. El *AssessmentManager* invoca al método *generateSuggestion* de la *ProcessArea*, en el cual se infiere la valoración sugerida en base a los datos de entrada y a las valoraciones de los objetivos asociados al área de proceso. Finalmente, la *ProcessArea* notifica a los interesados que ha generado una nueva valoración sugerida, pasando el nuevo valor dentro de un *ModelEvent*. La *ProcessAreaView* recibe la notificación y actualiza la interfaz con la nueva valoración sugerida.

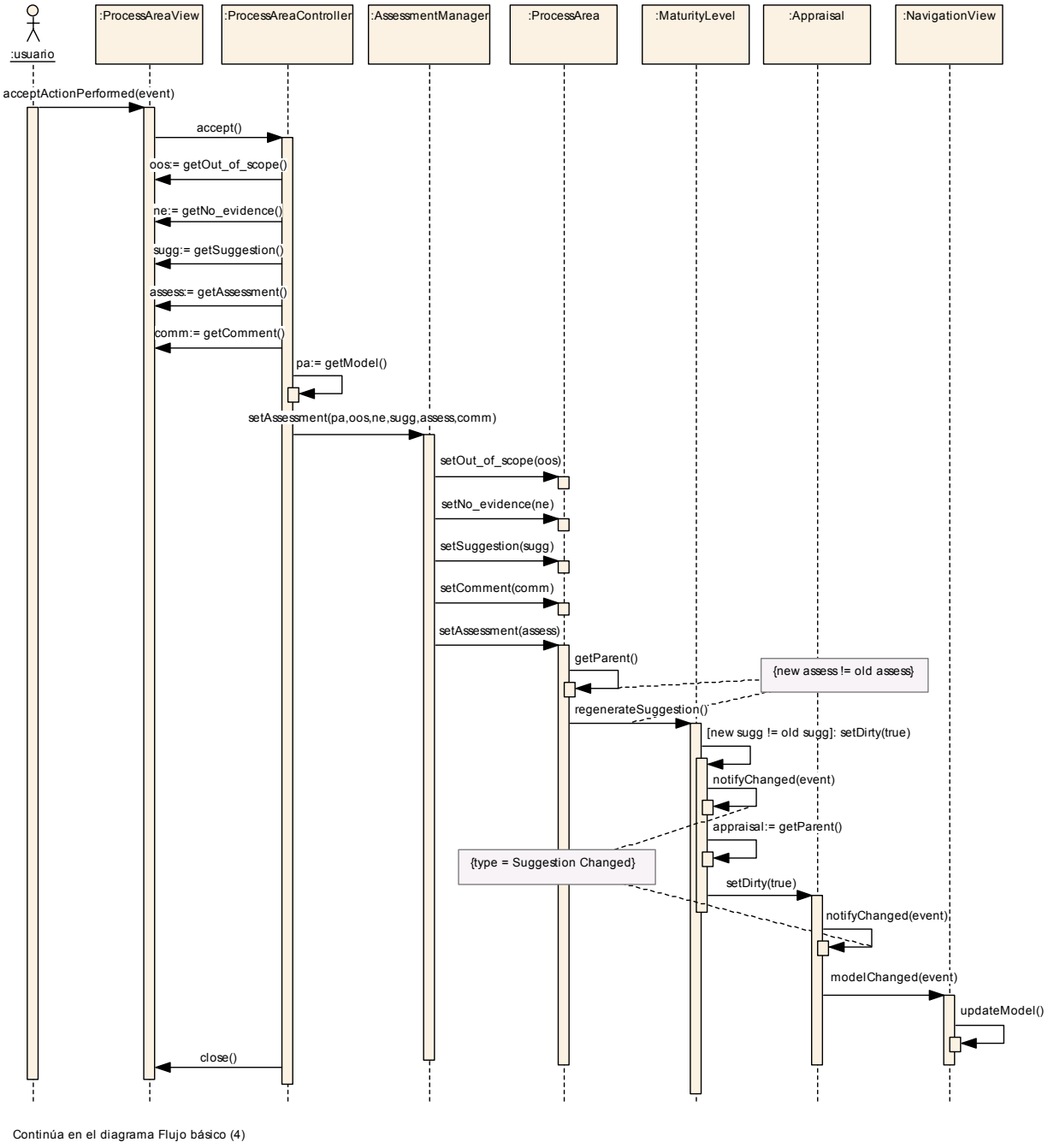

Figura 7.33c. Flujo básico (3). El usuario presiona el botón Aceptar en la ventana de evaluación, lo que se traduce en la invocación del método *acceptActionPerformed* sobre la *ProcessAreaView*. La vista invoca el método *accept* del *ProcessAreaController*, quien se encarga de recuperar todos los datos de la pantalla (*getOut_of_scope ... getComment*) y la *ProcessArea* asociada como modelo (*getModel*). A continuación, el *ProcessAreaController* instancia al *AssessmentManager* e invoca al método *setAssessment* pasándole los valores recuperados. El *AssessmentManager* setea los valores en la *ProcessArea* (*setOut_of_scope ... setAssessment*). El método *setAssessment* de *ProcessArea* verifica si la nueva evaluación difiere de la existente, en cuyo caso invoca al método *regenerateSuggestion* de su *parent* (*MaturityLevel*). El método *regenerateSuggestion* de *MaturityLevel* calcula el nuevo valor sugerido aplicando las reglas de SCAMPI, y en caso de que sea diferente del actual setea el indicador de necesidad de revisión (*setDirty(true)*), notifica a los interesados y propaga la indicación de revisión a su padre (objeto *Appraisal*). El *Appraisal* lanza una notificación que llega a la *NavigationView*, quien se

encarga de actualizar el árbol del modelo (*JTree*) para resaltar el nivel de madurez afectado por el cambio (*updateModel*). Finalmente, el *ProcessAreaController* cierra la ventana de evaluación (método *close()*).

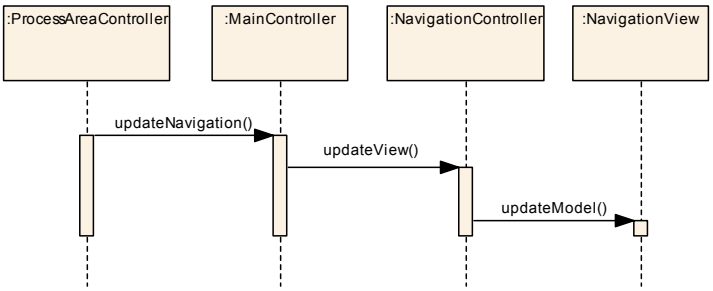

Figura 7.33d. Flujo básico (4). Luego de setear la nueva evaluación, el *ProcessAreaController* invoca al método *updateNavigation* del *MainController*, quien a su vez invoca al *updateView* del *NavigationController*. Este último, invoca al método *updateModel* de la *NavigationView* para que actualice el árbol del modelo.

Evaluar objetivos

El diagrama de la figura 7.34 muestra las clases que participan en la realización.

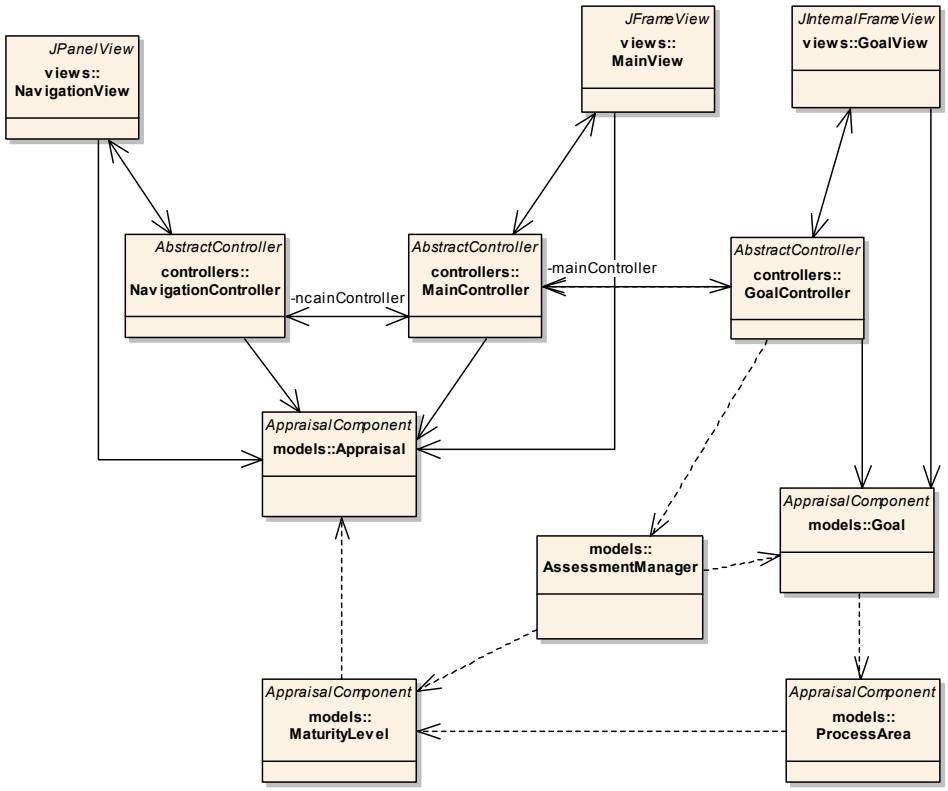

Figura 7.34. Participantes.

Los diagramas de las figuras 7.35a, 7.33b y 7.33c muestran las interacciones que se producen en la evaluación de un objetivo.

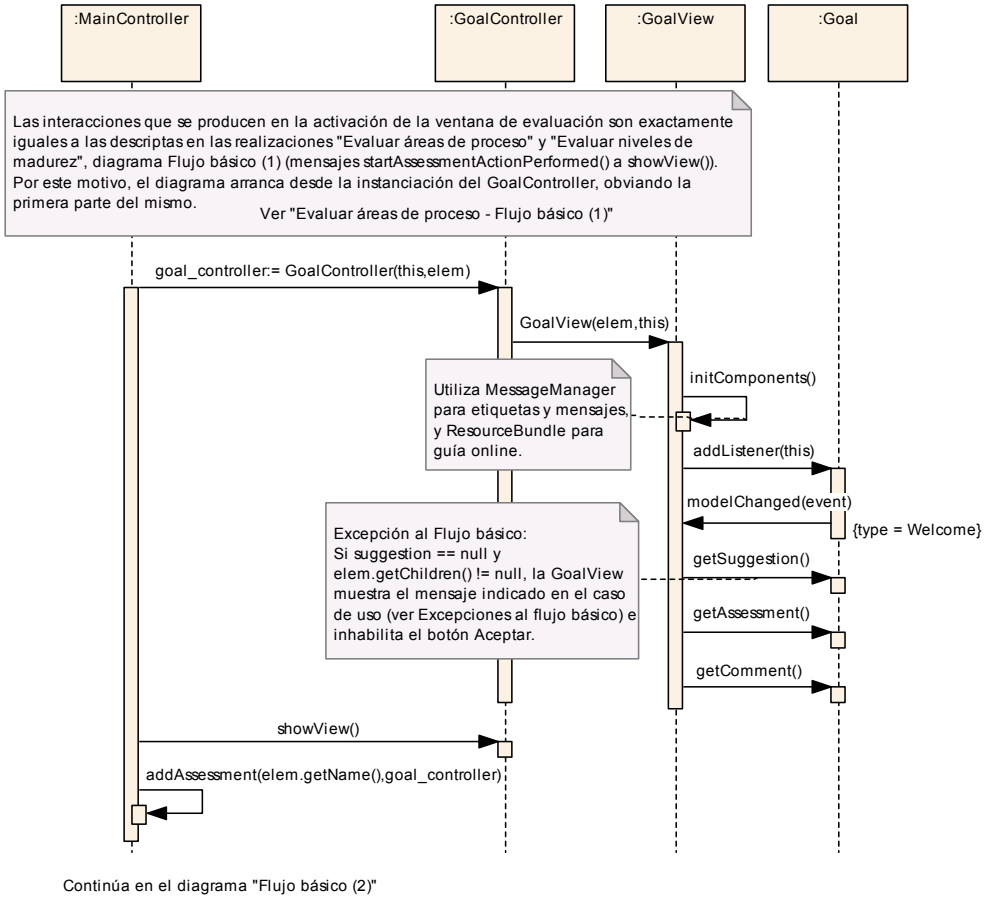

Figura 7.35a. Flujo básico (1). El usuario selecciona un objetivo en el árbol de navegación del modelo y a continuación selecciona la opción del menú "Evaluar". Esto desencadena las interacciones que se describieron en los diagramas anteriores (ver Nota). A continuación, el *MainController* instancia un *GoalController* pasándole como argumentos una referencia a sí mismo y una referencia al *Goal* que tiene como modelo (objeto *elem* de tipo *AppraisalComponent*). El *GoalController* instancia una *GoalView* pasándole una referencia al modelo (*Goal*) y otra a sí mismo. La *GoalView* inicializa la interfaz de usuario (*initComponents*), y se registra como *listener* de su modelo, quien le envía como respuesta una notificación de bienvenida (*modelChanged* con evento "*Welcome*"). Al recibir la notificación, la *GoalView* obtiene los atributos del objetivo (métodos *getSuggestion*, *getAssessment*, *getComment*) y los vuelca en los campos de la interfaz de usuario. Si no existe una valoración sugerida y el objetivo tiene hijos, la *GoalView* muestra un mensaje de advertencia al usuario (mensaje explicitado en el caso de uso, Excepción al flujo básico). Finalmente, el *MainController* agrega el *GoalController* instanciado al pool de controladores de evaluación, para registrarlo como evaluación en curso.

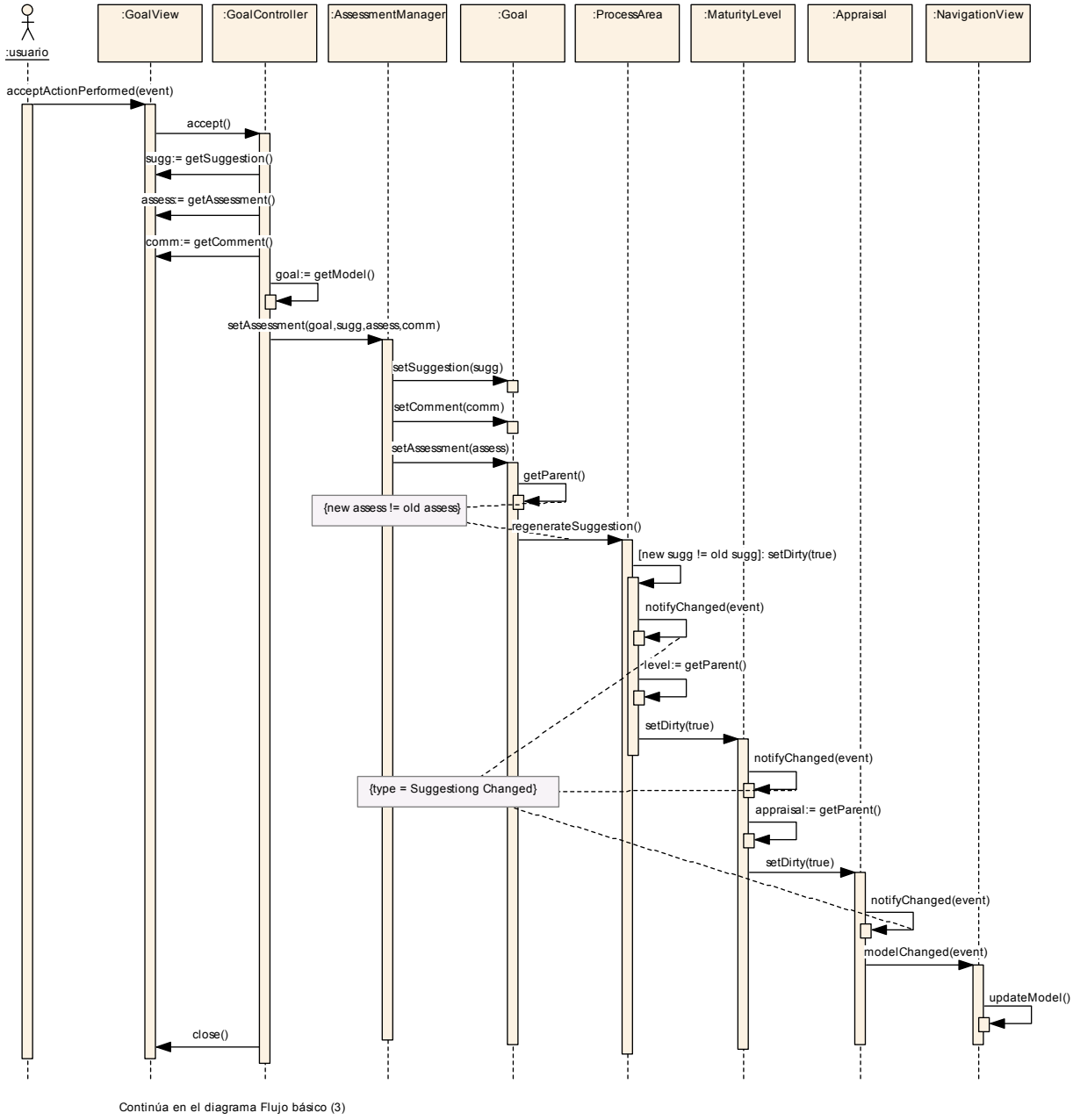

Figura 7.35b. Flujo básico (2). El usuario presiona el botón Aceptar en la ventana de evaluación, lo que se traduce en la invocación del método *acceptActionPerformed* sobre la *GoalView*. La vista invoca el método *accept* del *GoalController*, quien se encarga de recuperar todos los datos de la pantalla (*getSuggestion ... getComment*) y *Goal* asociado como modelo (*getModel*). A continuación, el *GoalController* instancia al *AssessmentManager* e invoca al método *setAssessment* pasándole los valores recuperados. El *AssessmentManager* setea los valores en el *Goal* (*setSuggestion ... setAssessment*). El método *setAssessment* de *Goal* verifica si la nueva evaluación difiere de la existente, en cuyo caso invoca al método *regenerateSuggestion* de su *parent* (*ProcessArea*). El método *regenerateSuggestion* de *ProcessArea* calcula el nuevo valor sugerido aplicando las reglas de SCAMPI, y en caso de que sea diferente del actual setea el indicador de necesidad de revisión (*setDirty(true)*), notifica a los interesados y propaga la indicación de revisión a su padre (objeto *MaturityLevel*). El *MaturityLevel* notifica a los interesados y propaga la indicación de revisión a su padre (objeto *Appraisal*). El *Appraisal* lanza una notificación que llega a la *NavigationView*, quien se encarga de actualizar el árbol del modelo (*JTree*)

para resaltar el nivel de madurez afectado por el cambio (*updateModel*). Finalmente, el *GoalController* cierra la ventana de evaluación (método *close()*).

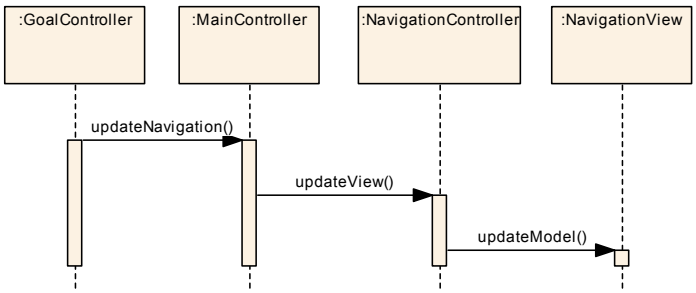

Figura 7.35c. Flujo básico (3). Luego de setear la nueva evaluación, el *GoalController* invoca al método *updateNavigation* del *MainController*, quien a su vez invoca al *updateView* del *NavigationController*. Este último, invoca al método *updateModel* de la *NavigationView* para que actualice el árbol del modelo.

Evaluar prácticas

El diagrama de la figura 7.36 muestra las clases que participan en la realización.

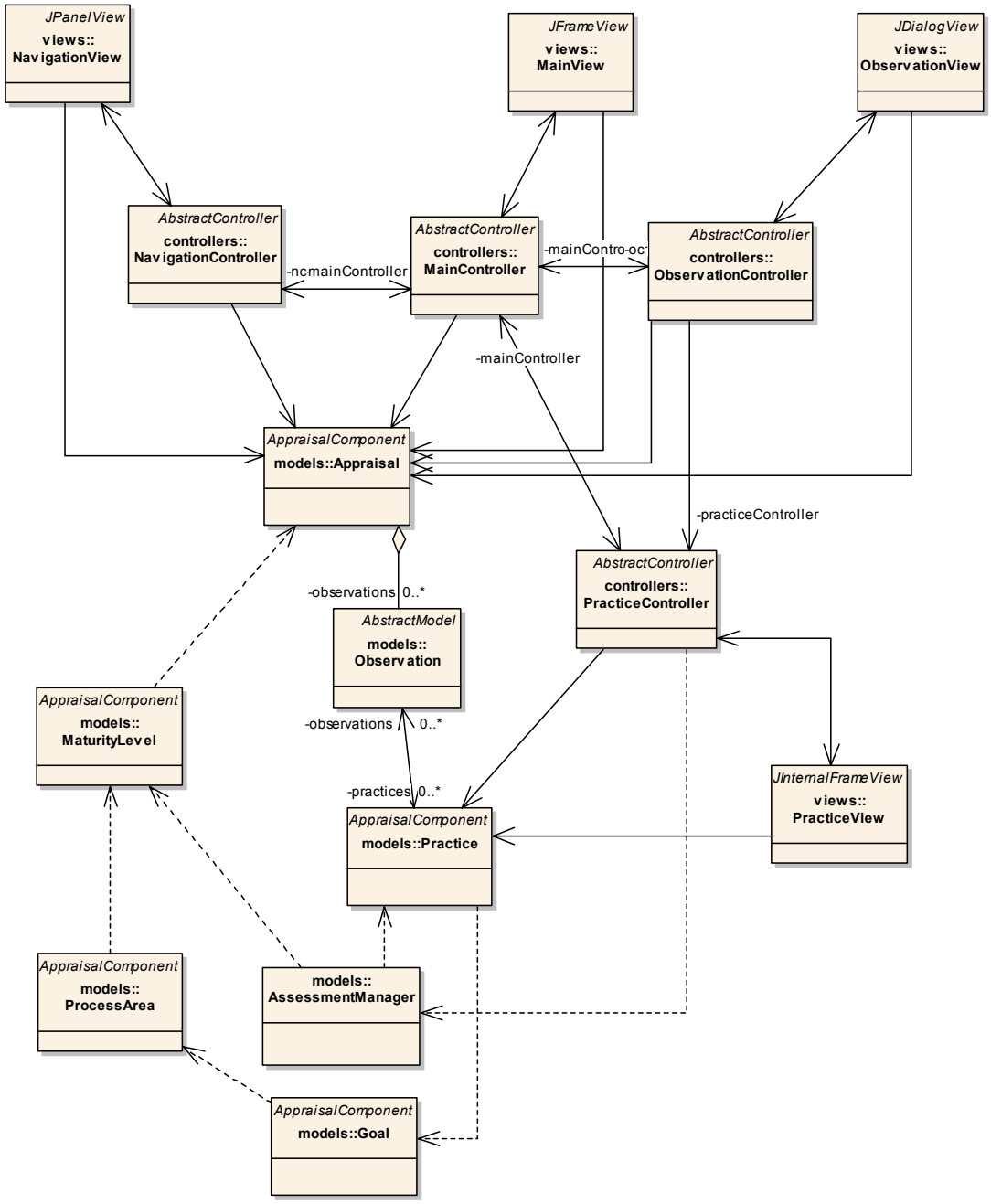

Figura 7.36. Participantes.

Los diagramas de las figuras 7.37a, 7.37b, 7.37c, 7.37d y 7.37e muestran las interacciones que se producen en la evaluación de una práctica.

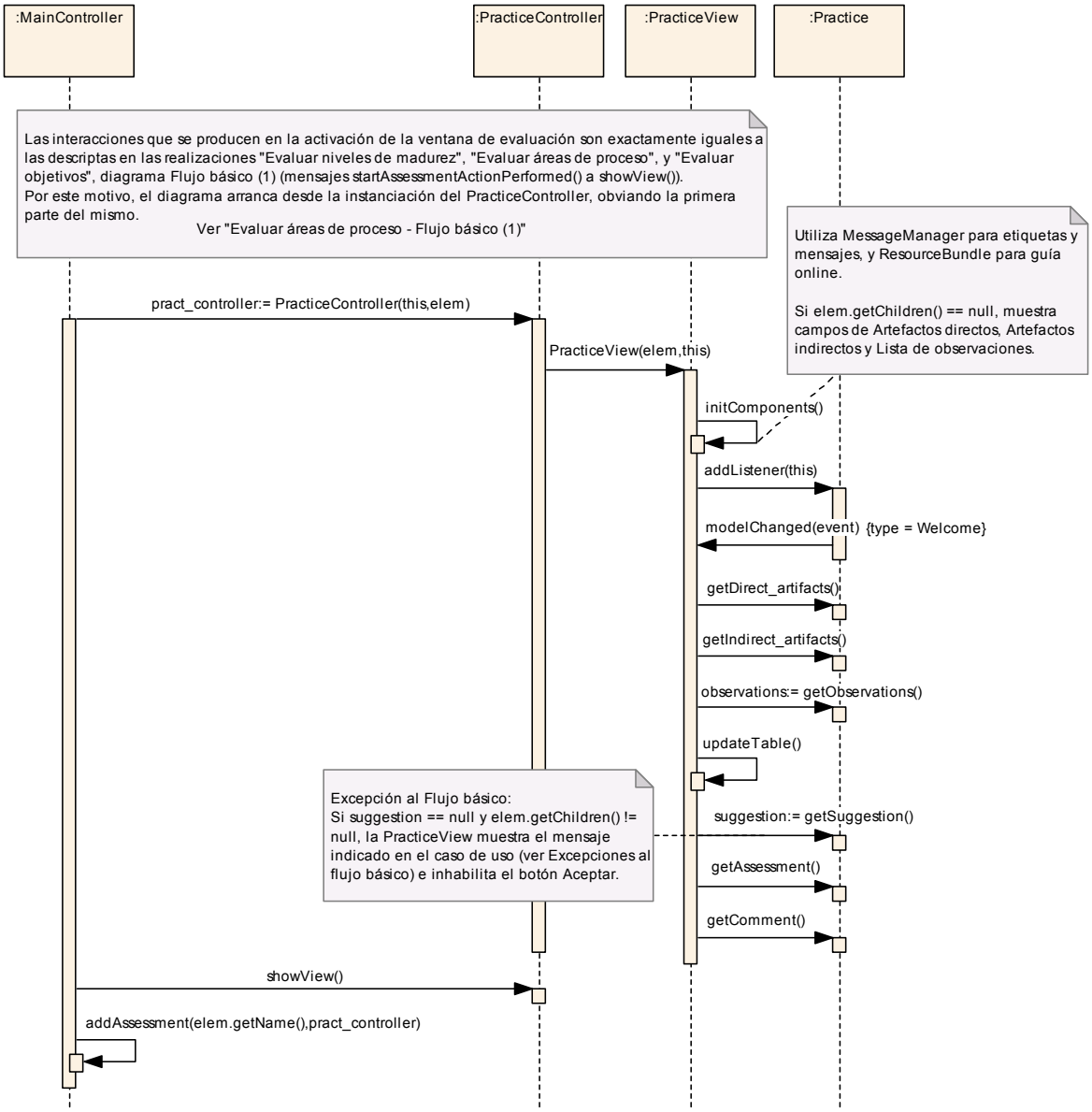

Figura 7.37a. Flujo básico (1). El usuario selecciona una práctica en el árbol de navegación del modelo y a continuación selecciona la opción del menú "Evaluar". Esto desencadena las interacciones que se describieron en los diagramas anteriores (ver Nota). A continuación, el *MainController* instancia un *PracticeController* pasándole como argumentos una referencia a sí mismo y una referencia a la *Practice* que tiene como modelo (objeto *elem* de tipo *AppraisalComponent*). El *PracticeController* instancia una *PracticeView* pasándole una referencia al modelo (*Practice*) y otra a sí mismo. La *PracticeView* inicializa la interfaz de usuario (*initComponents*), y se registra como listener de su modelo, quien le envía como respuesta una notificación de bienvenida (*modelChanged* con evento "*Welcome*"). Al recibir la notificación, la *PracticeView* obtiene los atributos de la práctica (métodos *getDirect_artifact ... getComment*) y los vuelca en los campos de la interfaz de usuario. Si no existe una valoración sugerida y la práctica tiene hijos, la *PracticeView* muestra un mensaje de advertencia al usuario (mensaje explicitado en el caso de uso, Excepción al flujo básico). Finalmente, el *MainController* da la orden de mostrar su

vista asociada al *PracticeController* y lo agrega al pool de controladores de evaluación, para registrarlo como evaluación en curso.

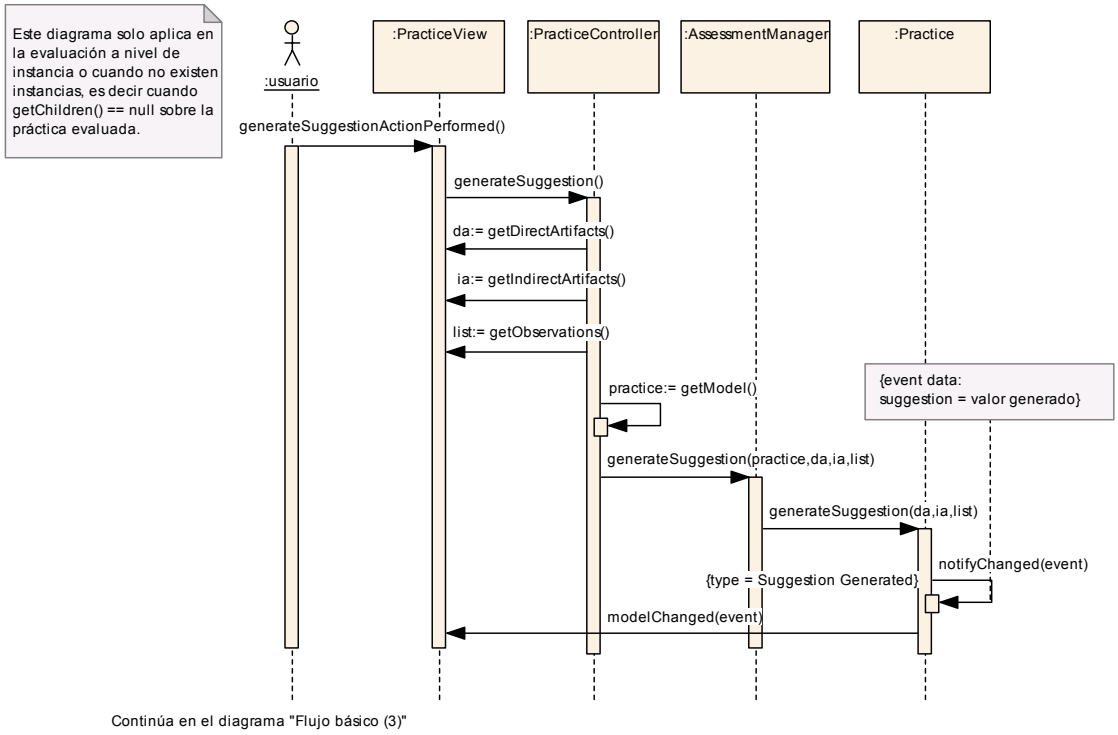

Figura 7.37b. Flujo básico (2). El usuario modifica los campos "Artefactos directos", "Artefactos indirectos" o la lista de "Observaciones" en la interfaz de evaluación, lo que se traduce en la invocación del método *generateSuggestionActionPerformed* en la *PracticeView*. La *PracticeView* invoca el método *generateSuggestion* del *PracticeController*, quien se encarga de recuperar los valores de la ventana y obtener su modelo asociado (objeto *Practice*). A continuación, instancia al *AssessmentManager* e invoca al método *generateSuggestion* pasándole como argumentos la *Practice* y los valores de la ventana. El *AssessmentManager* invoca al método *generateSuggestion* de la *Practice*, en el cual se infiere la valoración sugerida en base a los datos de entrada. Finalmente, la *Practice* notifica a los interesados que ha generado una nueva valoración sugerida, pasando el nuevo valor dentro de un *ModelEvent*. La *PracticeView* recibe la notificación y actualiza la interfaz con la nueva valoración sugerida.

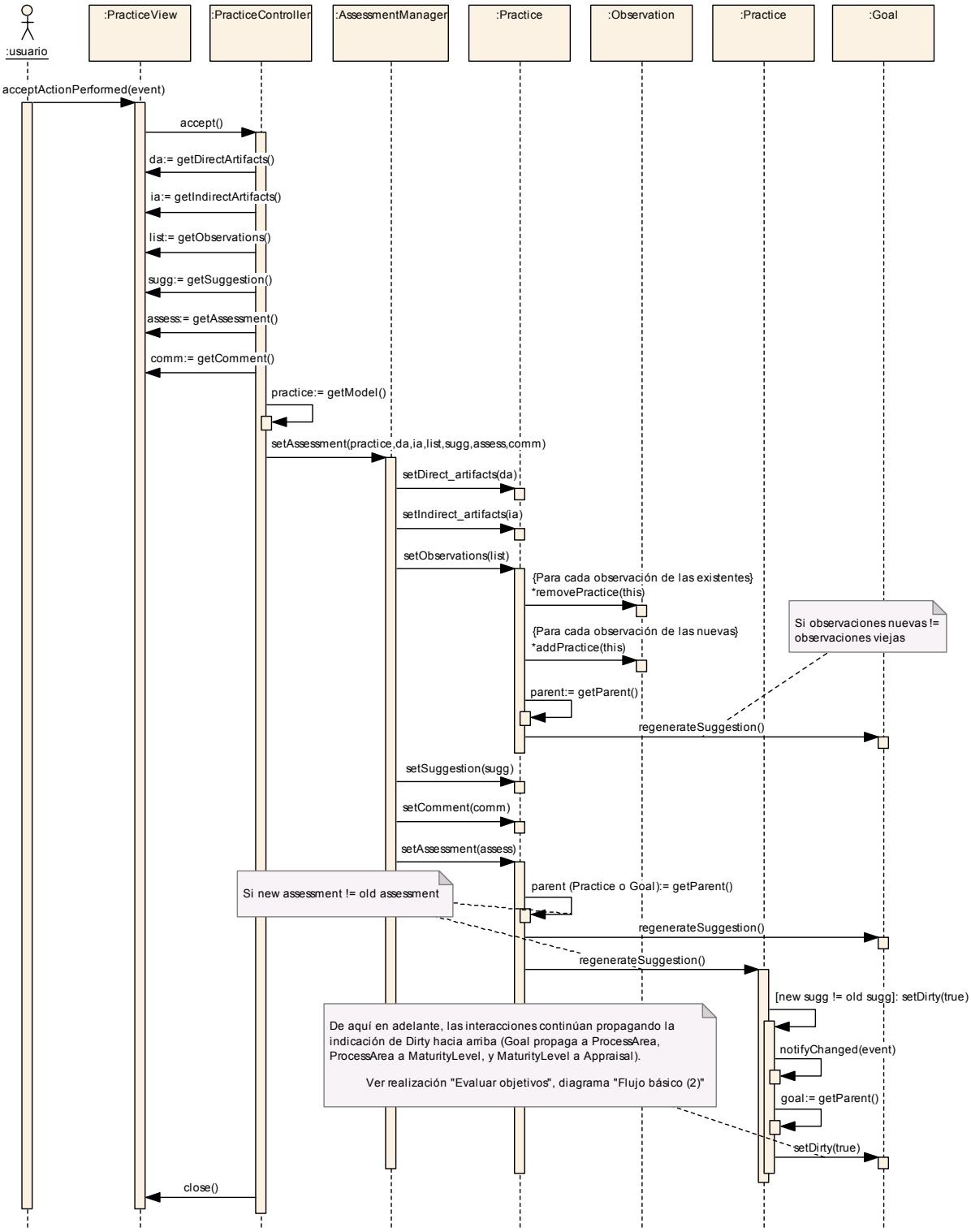

Figura 7.37c. Flujo básico (3). El usuario presiona el botón Aceptar en la ventana de evaluación, lo que se traduce en la invocación del método *acceptActionPerformed* sobre la *PracticeView*. La vista invoca el método *accept* del *PracticeController*, quien se encarga de recuperar todos los datos de la pantalla (*getDirectArtifacts ... getComment*) y la *Practice* asociada como modelo (*getModel*). A continuación, el *PracticeController* instancia al *AssessmentManager* e invoca al método *setAssessment* pasándole los valores recuperados. El *AssessmentManager* setea los valores en la *Practice* (*setDirect_artifacts ... setComment*). El método *setObservations* de *Practice* elimina todas las referencias existentes y crea las nuevas en cada uno de los objetos *Observation* vinculados con la práctica. Además, en caso de que las observaciones nuevas sean distintas a las existentes, invoca al método *regenerateSuggestion* del *parent*.

El método *setAssessment* de *Practice* verifica si la nueva evaluación difiere de la existente, en cuyo caso invoca al método *regenerateSuggestion* de su *parent* (que puede ser una *Practice* o un *Goal*). El método *regenerateSuggestion* (tanto de *Practice* como de *Goal*) calcula el nuevo valor sugerido aplicando las reglas de SCAMPI, y en caso de que sea diferente del actual setea el indicador de necesidad de revisión (*setDirty(true)*), notifica a los interesados y propaga la indicación de revisión a su padre. Las indicaciones de revisión se continúan propagando hacia los padres hasta que llegan al objeto *Appraisal*. El *Appraisal* lanza una notificación que llega a la *NavigationView*, quien se encarga de actualizar el árbol del modelo (*JTree*) para resaltar el nivel de madurez afectado por el cambio (*updateModel*). Finalmente, el *PracticeController* cierra la ventana de evaluación (método *close()*).

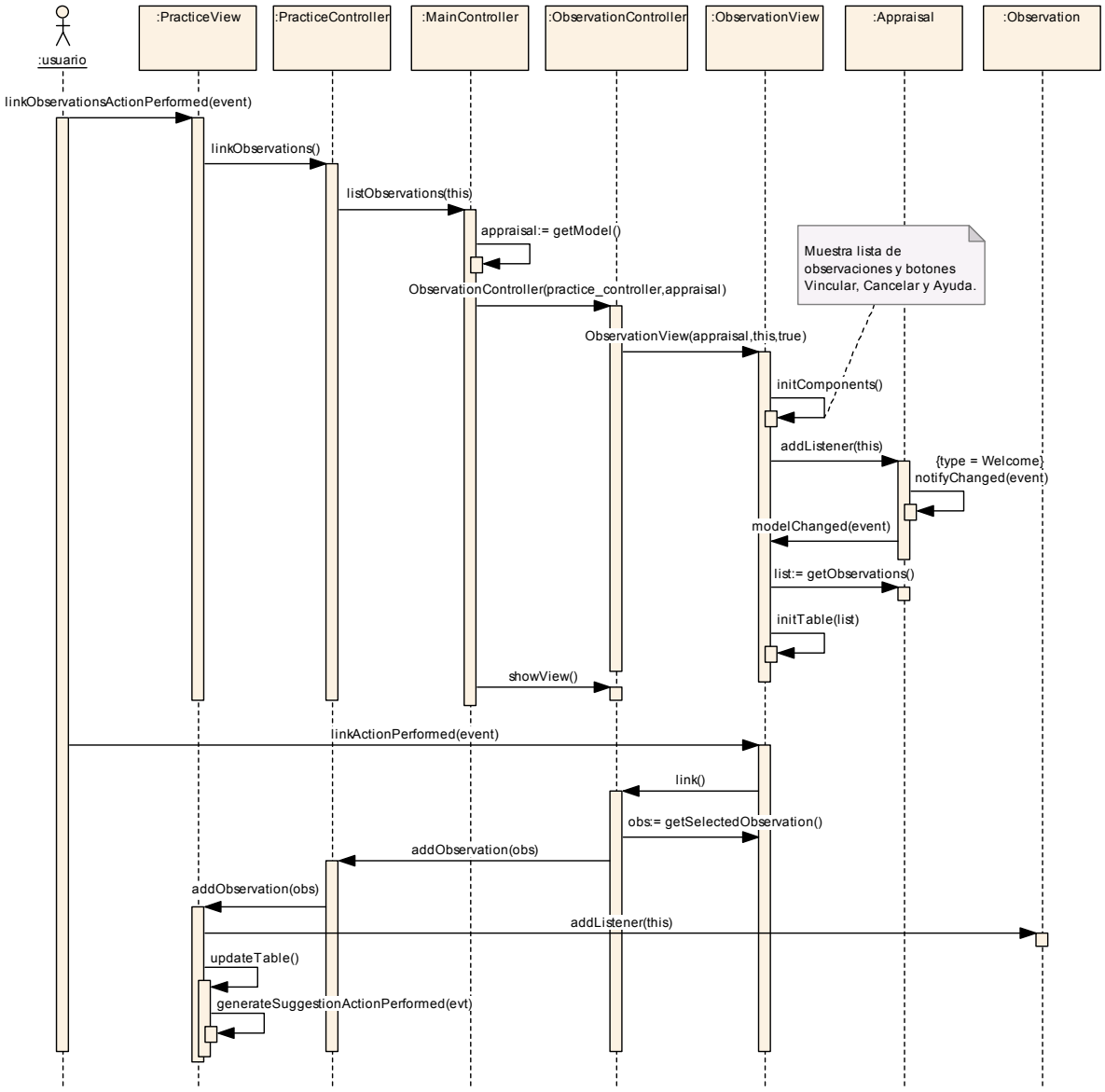

Figura 7.37d. Vinculación de observaciones. El usuario presiona el botón "Vincular observaciones" en la pantalla de evaluación de prácticas, lo que se traduce en la invocación del método *linkObservationsActionPerformed* de la *PracticeView*. La vista invoca al método *linkObservations* de su controlador asociado (*PracticeController*). El *PracticeController* invoca al método *listObservations* del *MainController* pasándole una referencia a sí mismo. El *MainController* crea una instancia del *ObservationController* pasándole la referencia al *PracticeController* y su modelo asociado (*Appraisal*). El *ObservationController* crea la *ObservationView* pasándole el *Appraisal* y una referencia a sí mismo, y un indicador de vinculación, ante lo cual la vista muestra una interfaz recortada al usuario (lista de

observaciones y botón para Vincular). La *ObservationView* se registra como listener del *Appraisal*, ante lo cual recibe una notificación de cambio del tipo "*Welcome*". Al recibir la notificación, obtiene la lista de observaciones contenidas en el *Appraisal* (método *getObservations*) e inicializa la tabla en pantalla con las observaciones obtenidas. Cuando el usuario selecciona algunas observaciones de la lista y presiona el botón Vincular, se invoca el método *linkActionPerformed* de la *ObservationView*. La *ObservationView* invoca al método link del *ObservationController*, quien se encarga de recuperar la observación seleccionada en la pantalla (método *getSelectedObservation*) y pasársela al *PracticeController* mediante el método *addObservation*. El *PracticeController* invoca al método *addObservation* de la *PracticeView* pasándole la observación recibida. La *PracticeView* se registra como listener de la observación de manera de recibir las notificaciones de los cambios, actualiza la tabla de observaciones (lo que a su vez provoca la regeneración de la valoración sugerida).

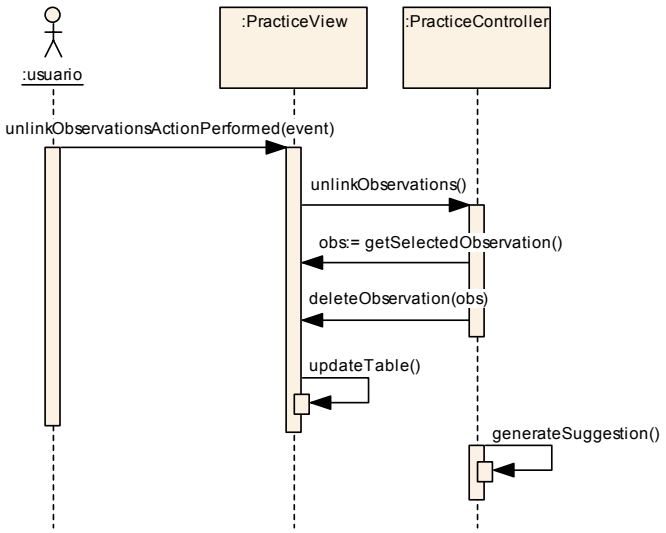

Figura 7.37d. Desvinculación de observaciones. El usuario selecciona una observación de la lista y presiona el botón "Desvincular", lo que se traduce en la invocación del método *unlinkObservationsActionPerformed* de la *PracticeView*. La *PracticeView* invoca al método *unlink* del *PracticeController*, quien se encarga de obtener el índice de la observación seleccionada (método *getSelectedObservation*) y de invocar al método *deleteObservation* de la *PracticeView* para que la elimine de su lista. Finalmente, invoca al método *generateSuggestion*.

Administrar observaciones

El diagrama de la figura 7.38 muestra las clases que participan en la realización.

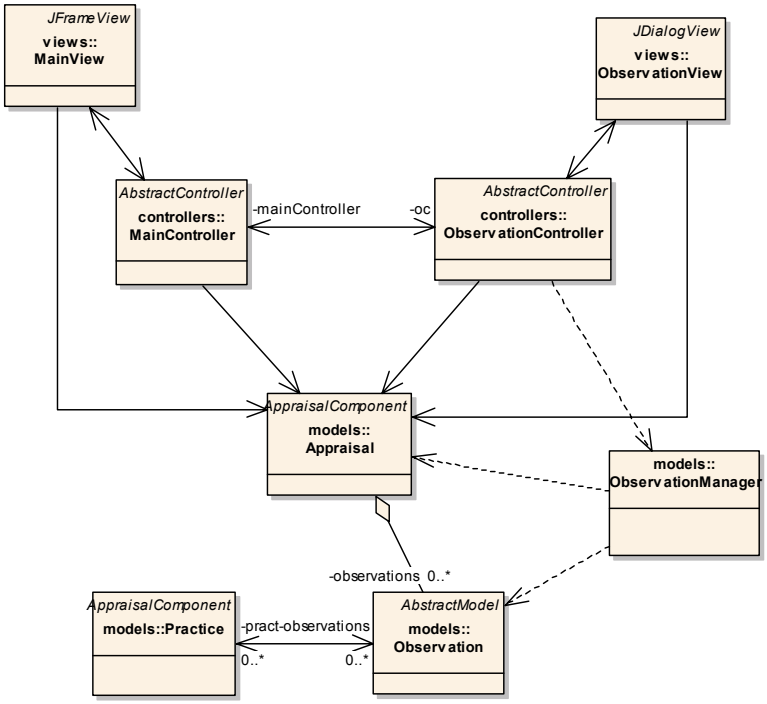

Figura 7.38. Participantes.

Los diagramas de las figuras 7.39a, 7.39b, 7.39c y 7.39d muestran las interacciones que se producen en la administración de observaciones.

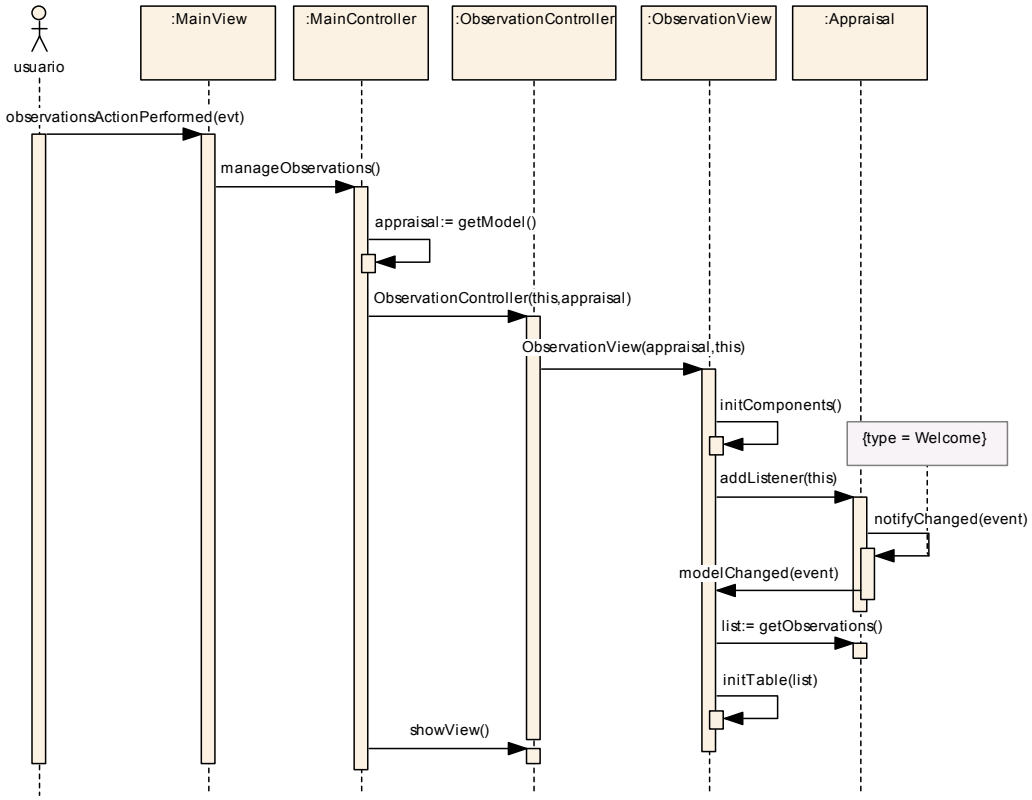

Figura 7.39a. Flujo básico. El usuario selecciona la opción de menú "Obvservaciones", lo que se traduce en la invocación del método *observationsActionPerformed* de la *MainView*. La *MainView* delega la

invocación en su controlador (*MainController*), mediante el método *manageObservations*. El *MainController* obtiene su modelo (objeto *Appraisal*) y crea una instancia del *ObservationController* pasándole una referencia a sí mismo y una referencia al modelo (*appraisal*). El *ObservationController* instancia la *ObservationView* pasándole una referencia a sí mismo y la referencia al *Appraisal* que tiene como modelo. La *ObservationView* inicializa la interfaz de usuario (*initComponents*) y se registra como *listener* del modelo. El *Appraisal* (modelo) le da la bienvenida mediante un *ModelEvent* tipo "*Welcome*". Al recibir la notificación, la *ObservationView* recupera la lista de observaciones del *Appraisal* e inicializa la tabla donde se listan las observaciones existentes.

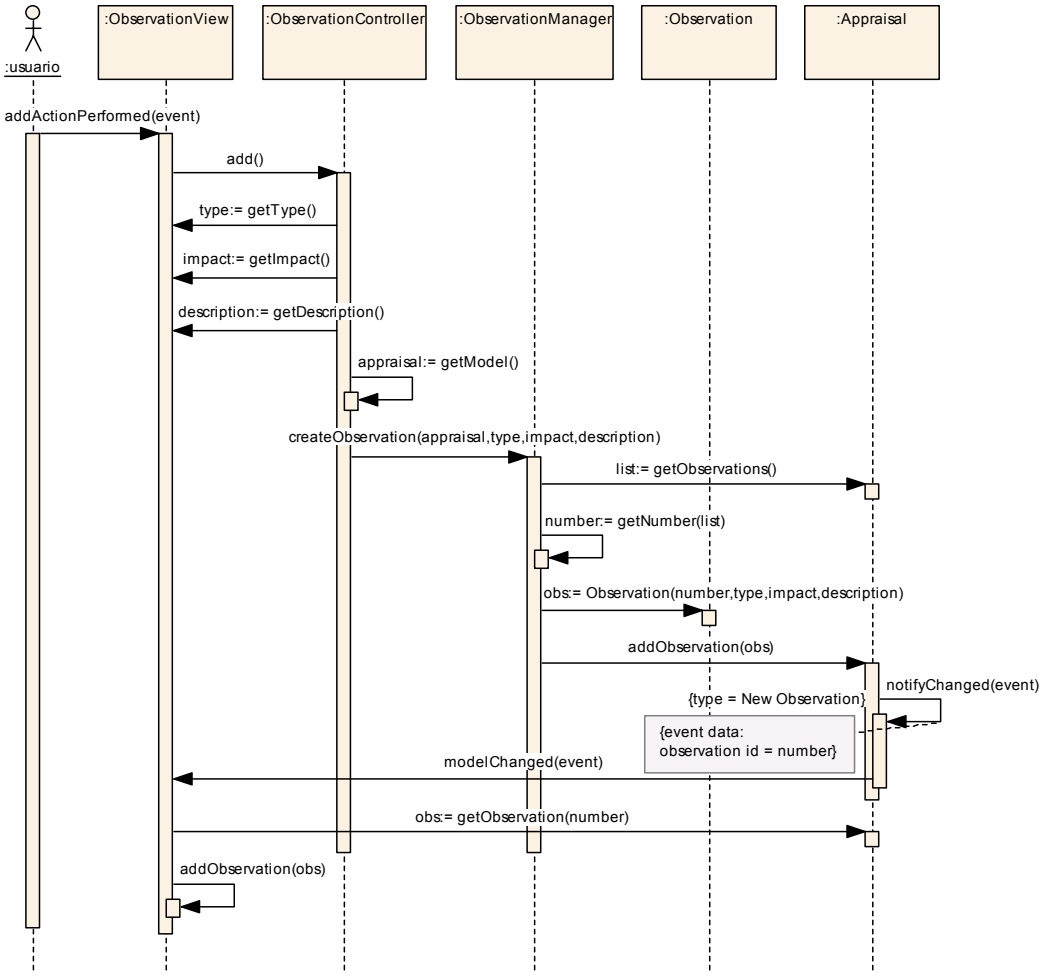

Figura 7.39b. Flujo alternativo - Crear observación. El usuario ingresa los datos de una nueva observación y presiona el botón "Agregar", lo que se traduce en la invocación del método *addActionPerformed* de la *ObservationView*. La vista invoca al método add de su controlador (*ObservationController*), quien se encarga de recuperar los datos de la pantalla (*getType ... getDescription*), recuperar su modelo asociado (*Appraisal*), e invocar al método *createObservation* del *ObservationManager*, pasándole todos los datos como argumentos. El *ObservationManager* obtiene la lista de observaciones mediante el método *getObservations* del *Appraisal*, y genera un nuevo número de observación en base a los existentes (máximo número + 1). A continuación, crea un objeto *Observation* con los datos recibidos y lo agrega a la lista de observaciones de la evaluación mediante el método *addObservation* del *Appraisal*. La invocación a *addObservation* genera que el objeto *Appraisal* notifique a sus interesados que se ha agregado una nueva observación, mediante un *ModelEvent*. Entre los interesados se encuentra la *ObservationView*, quien al recibir la notificación recupera el objeto

Observation (*getObservation(number)*) y lo agrega a la lista que muestra al usuario (método *addObservation* de la *ObservationView*).

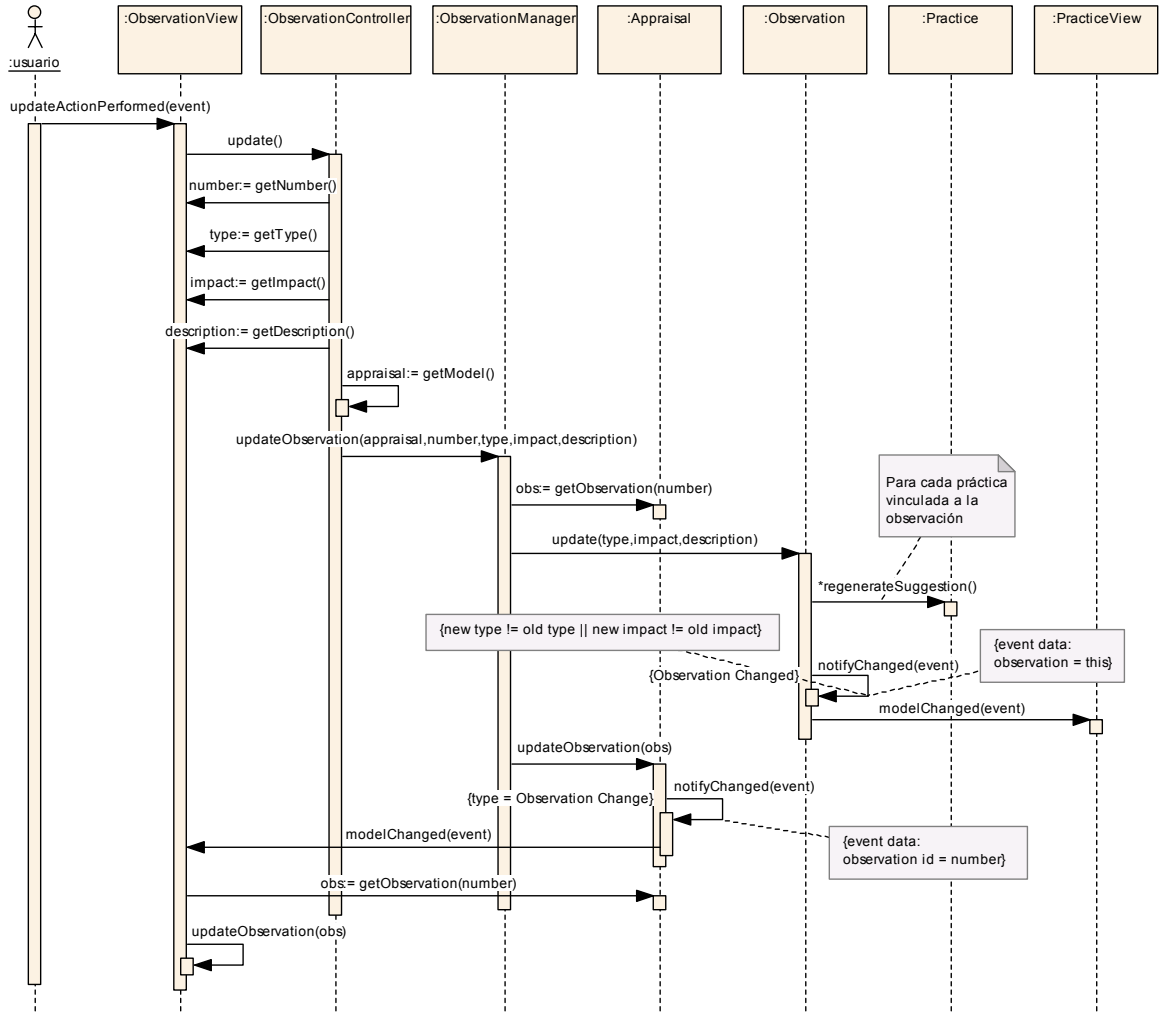

Figura 7.39c. Flujo alternativo - Modificar observación. El usuario selecciona una observación de la lista en pantalla, modifica sus datos y presiona el botón "Modificar", lo que se traduce en la invocación del método *updateActionPerformed* de la *ObservationView*. La vista invoca al método *update* de su controlador (*ObservationController*), quien se encarga de recuperar los datos de la pantalla (*getNumber ... getDescription*), recuperar su modelo asociado (*Appraisal*), e invocar al método *updateObservation* del *ObservationManager*, pasándole todos los datos como argumentos. El *ObservationManager* obtiene la observación a modificar mediante el método *getObservation* del *Appraisal*, y la actualiza mediante el método *update* de *Observation*. El método update verifica si se han cambiado los atributos *type* o *impact*, en cuyo caso invoca al método *notifyChange* quien se encarga de invocar al método *regenerateSuggestion* de todas las prácticas que tienen vinculación con la observación modificada. Finalmente, el *ObservationManager* actualiza la observación en la lista de observaciones del *Appraisal* mediante el método *updateObservation* de la clase *Appraisal*. La invocación a *updateObservation* genera que el objeto *Appraisal* notifique a sus interesados que se ha modificado observación, mediante un *ModelEvent*. Entre los interesados se encuentra la *ObservationView*, quien al recibir la notificación recupera el objeto *Observation* (*getObservation(number)*) y lo actualiza en la lista que muestra al usuario (método *updateObservation* de la *ObservationView*).

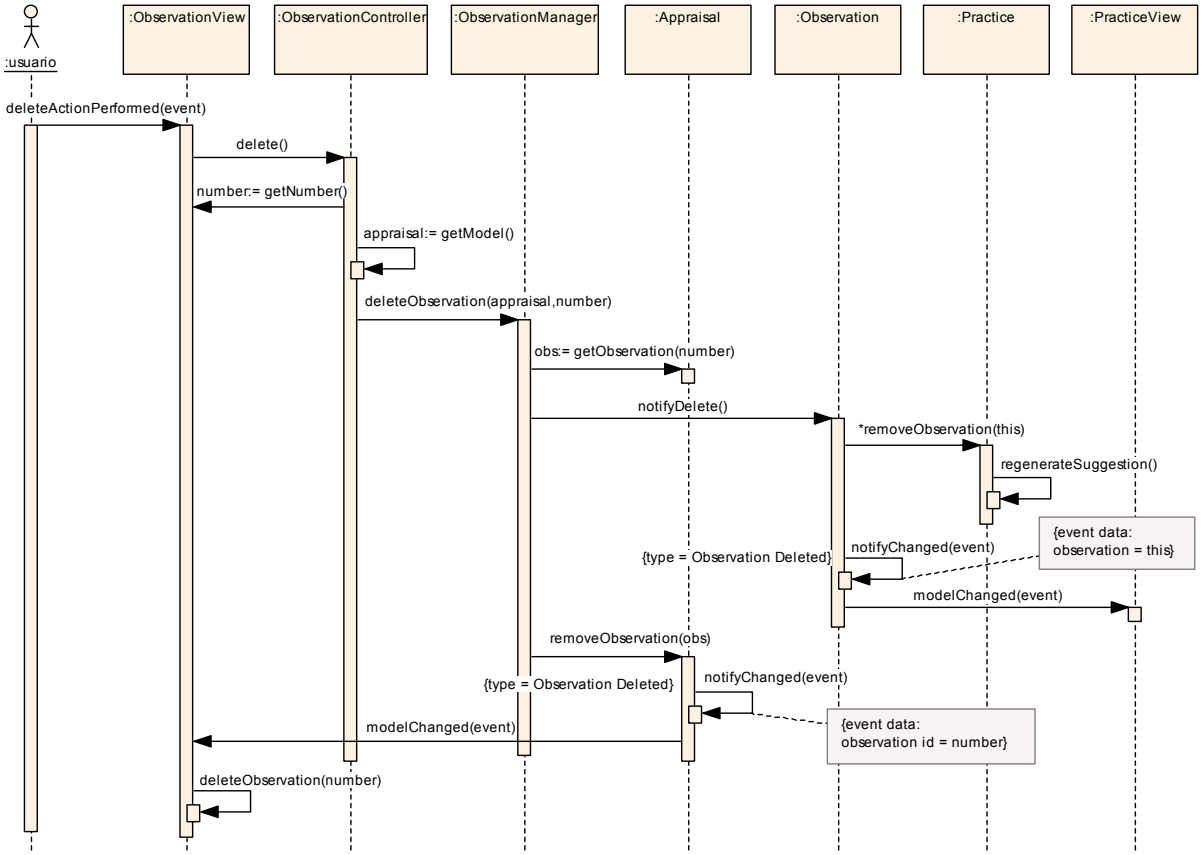

Figura 7.39d. Flujo alternativo - Eliminar observación. El usuario selecciona una observación de la lista en pantalla y presiona el botón "Eliminar", lo que se traduce en la invocación del método *deleteActionPerformed* de la *ObservationView*. La vista invoca al método *delete* de su controlador (*ObservationController*), quien se encarga de presentar una ventana de confirmación (*JOptionPane*) y en caso que el usuario confirme, recupera los datos de la pantalla (*getNumber*), recupera su modelo asociado (*Appraisal*), e invoca al método *deleteObservation* del *ObservationManager*, pasándole el *Appraisal* y el número de observación como argumentos. El *ObservationManager* obtiene la observación a eliminar mediante el método *getObservation* del *Appraisal* e invoca al método *notifyDelete* quien se encarga de invocar al método *removeObservation* de todas las prácticas que tienen vinculación con la observación eliminada, lo que a su vez genera una invocación a *regenerateSuggestion* dentro de cada práctica. Finalmente, el *ObservationManager* elimina la observación de la lista de observaciones del *Appraisal* mediante el método *removeObservation* de la clase *Appraisal*. La invocación a *removeObservation* genera que el objeto *Appraisal* notifique a sus interesados que se ha eliminado una observación, mediante un *ModelEvent*. Entre los interesados se encuentra la *ObservationView*, quien al recibir la notificación elimina la observación de la lista que muestra al usuario (método *deleteObservation* de la *ObservationView*).

Generar reporte

El diagrama de la figura 7.40 muestra las clases que participan en la realización.

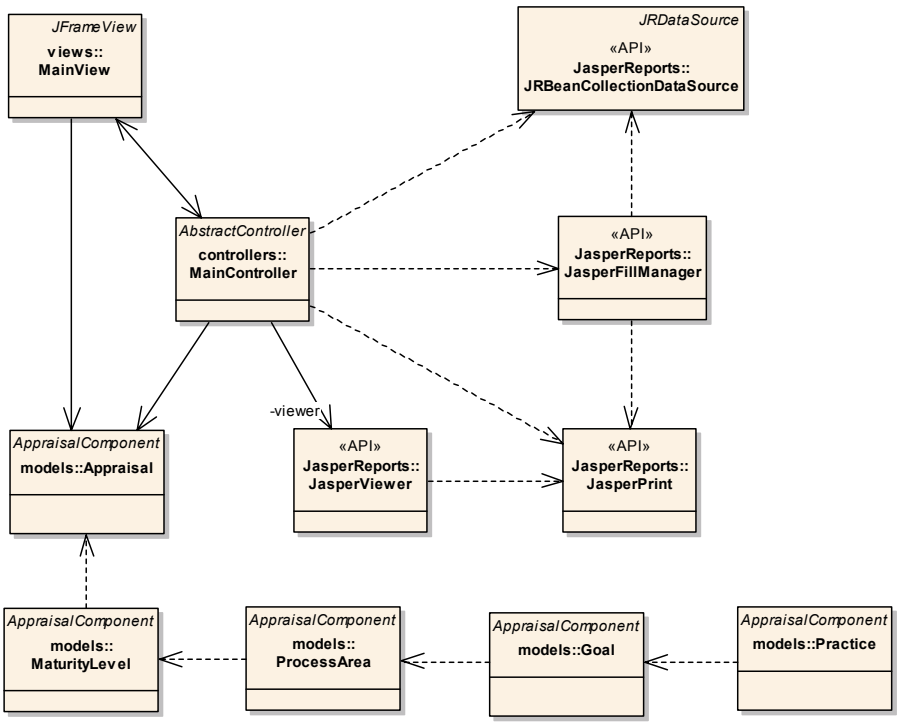

Figura 7.40. Participantes.

Los diagramas de las figuras 7.41a y 7.41b muestran las interacciones que se producen en la generación de un reporte.

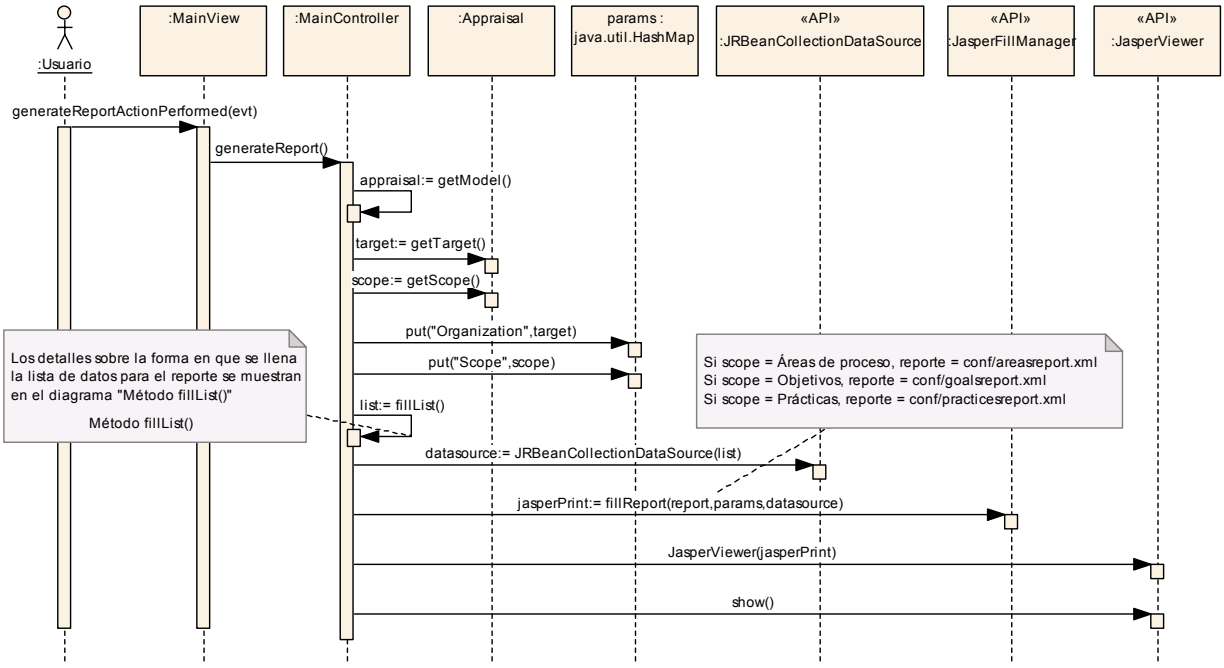

Figura 7.41a. Flujo básico. El usuario selecciona la opción de menú "Generar reporte", lo que se traduce en la invocación del método *generateReportActionPerformed* de la *MainView*. La *MainView* delega la invocación en su controlador (*MainController*), mediante el método *generateReport*. El *MainController* obtiene su modelo (objeto *Appraisal*), y los parámetros *target* y *scope* del mismo (Organización bajo

evaluación y Alcance de la evaluación respectivamente). Con esos parámetros llena un *HashMap* (parámetros generales del reporte). A continuación invoca al método *fillList()* para llenar la colección de datos de entrada del reporte (*list*) y construye un objeto *JRBeanCollectionDataSource* con la lista obtenida. Luego de eso utiliza el método *fillReport* de la clase *JasperFillManager*, pasándole la plantilla del reporte (*areasreport.jasper*, *goalsreport.jasper* o *practicesreport.jasper*), los parámetros generales (*HashMap*) y el *datasource* recién construído. El método *fillReport* retorna un objeto *JasperPrint*. Finalmente, el *MainController* crea un objeto *JasperViewer* pasándole el *JasperPrint* como parámetro, e invoca al método *show()* para que se muestre el reporte al usuario.

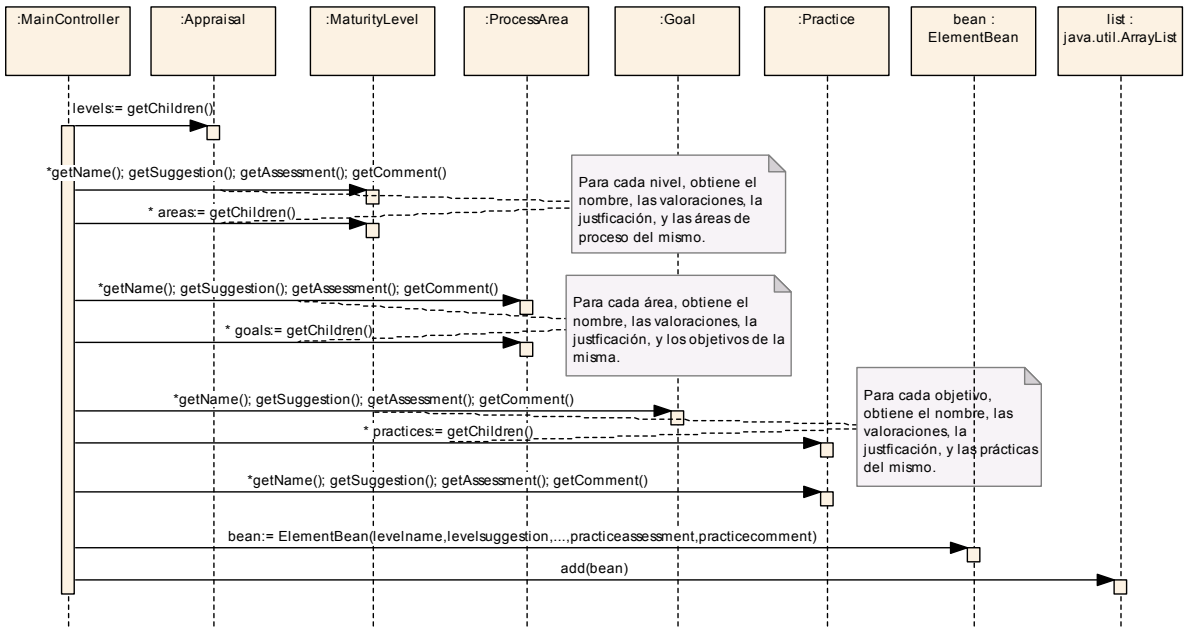

Figura 7.41b. : Método fillList().El método *fillList* construye una lista de beans con los datos de todos los elementos que intervienen en el reporte. Comienza obteniendo los niveles de la evaluación (invocación a *getChildren* sobre el *Appraisal*). Para cada nivel, obtiene los datos de evaluación y las áreas de proceso asociadas al mismo. Para cada área de proceso, obtiene los datos de evaluación y los objetivos asociados a la misma. Para cada objetivo, obtiene los datos de evaluación y las prácticas asociadas al mismo. Para cada práctica, obtiene los datos de evaluación. Finalmente, con todos los datos obtenidos inicializa un objeto *ElementBean* y lo agrega a la lista de resultados a retornar (*list*).

7.7 Verificación y análisis de consistencia

7.7.1 Verificación de los modelos

Se verificó la calidad de los distintos modelos por separado, a fines de garantizar que eran adecuados de acuerdo a la técnica seguida para su elaboración. Como resultado de la verificación, se efectuaron correcciones en el Modelo de clases de diseño y en las Realizaciones de diseño.

7.7.2 Resultado del análisis de consistencia

Se comprobó la coherencia entre los distintos modelos de acuerdo a las trazabilidades presentadas en el apartado 7.1 (Modelos y trazabilidad).

La comprobación se llevó a cabo mediante un conjunto de matrices de trazabilidad, generadas desde la herramienta *Enterprise Architect*.

Realizaciones de diseño vs Modelo de casos de uso

La matriz de la tabla 7.1 muestra en las filas las colaboraciones del modelo Realizaciones de diseño y en las columnas los casos de uso del Modelo de casos de uso. Como puede verse en la misma, cada uno de los casos de uso ha sido realizado en una colaboración.

	Administrar evaluaciones	Administrar observaciones	Evaluar niveles de madurez	Evaluar objetivos	Evaluar prácticas	Evaluar áreas de proceso	Generar reporte
Administrar evaluaciones	X						
Administrar observaciones		X					
Evaluar niveles de madurez			X				
Evaluar objetivos				X			
Evaluar prácticas					X		
Evaluar áreas de proceso						X	
Generar reporte							X

Tabla 7.1. Realizaciones de diseño vs. Modelo de casos de uso.

Modelo de clases de diseño vs Realizaciones de diseño

La matriz de la tabla 7.2 muestra en las filas las clases específicas de la aplicación (paquetes "*views*", "*controllers*" y "*models*") del Modelo de clases de diseño, y en las columnas las colaboraciones del modelo Realizaciones de diseño. Como puede verse en la misma, cada clase de diseño ha participado en al menos una colaboración. Las clases que no se encuentran marcadas en la matriz son clases abstractas, clases internas a algún subsistema o clases pertenecientes a la plataforma J2SE.

	Administrar evaluaciones	Administrar observaciones	Evaluar niveles de madurez	Evaluar objetivos	Evaluar prácticas	Evaluar áreas de proceso	Generar reporte
Appraisal	X	X	X	X	X	X	X
AppraisalFactory	X		X				
AppraisalManager	X		X				
AssessmentManager			X	X	X	X	
ComboObject		X		X	X	X	
Goal			X	X	X		X
GoalController				X			
GoalView				X			
HelpManager	X	X	X	X	X	X	X
InitAppraisalController	X						
InitAppraisalView	X						
JasperFillManager							X
JasperPrint							X
JasperViewer							X
JRBeanCollectionDataSource							X
MainController	X	X	X	X	X	X	X
MainView	X	X	X	X	X	X	X
MaturityLevel	X		X	X	X	X	X
MaturityLevelController			X				
MaturityLevelView			X				
ModelEvent	X						
NavigationController	X		X	X	X	X	
NavigationView	X		X	X	X	X	
Observation		X			X		
ObservationController		X			X		
ObservationManager		X					
ObservationView		X			X		
PersistenceManager	X						
Practice		X	X		X		X
PracticeController					X		
PracticeView					X		
ProcessArea			X	X	X	X	X
ProcessAreaController						X	
ProcessAreaView						X	
SystemException	X						

Tabla 7.2. Modelo de clases de diseño vs. Realizaciones de diseño.

Además de estas comprobaciones, se verificó que cada una de las operaciones de las clases de diseño corresponda a un mensaje de los diagramas de secuencia. No se grafica esa matriz debido a que su extensión es excesiva.

Subsistemas de soporte vs Realizaciones de diseño

La matriz de la tabla 7.3 muestra en las filas los subsistemas de soporte del modelo de diseño, y en las columnas las colaboraciones del modelo Realizaciones de diseño. Como puede verse en la misma, cada subsistema ha participado en al menos una colaboración.

	Administrar evaluaciones	Administrar observaciones	Evaluar niveles de madurez	Evaluar objetivos	Evaluar prácticas	Evaluar áreas de proceso	Generar reporte
JasperReports							X
JavaHelp	X	X	X	X	X	X	
messages	X	X	X	X	X	X	X
persistence	X						

Tabla 7.3. Subsistemas de soporte vs. Realizaciones de diseño.

Modelo de clases de diseño vs Modelo de clases de análisis

Las matriz de la tabla 7.4 muestra en las filas las clases específicas de la aplicación (paquetes "*views*", "*controllers*" y "*models*") correspondientes al Modelo de clases de diseño, y en las columnas las clases del Modelo de clases de análisis. Como puede verse en las mismas, cada clase de análisis se ha convertido en una o más clases de diseño. Las clases que no se encuentran marcadas en la matriz son clases abstractas, clases internas a algún subsistema o clases pertenecientes a la plataforma J2SE.

	AreaProceso	CatalogoGuias	ControladorEvaluaciones	ControladorObservaciones	ControladorValoraciones	Evaluacion	IArchivos	IAreaProceso	IConfirmacion	IInicioEvaluacion	INavegacion	INivelMadurez	IObjetivo	IObservaciones	IPractica	IPrincipal	IReporte	NivelMadurez	Objetivo	Observacion	Practica
Appraisal						X															
AppraisalFactory			X																		
AppraisalManager			X																		
AssessmentManager					X																
Goal																			X		
GoalController													X								
GoalView													X								
HelpManager		X																			
InitAppraisalController										X											
InitAppraisalView										X											
JasperViewer																	X				
MainController																X	X				
MainView																X					
MaturityLevel																		X			
MaturityLevelController												X									
MaturityLevelView												X									
NavigationController											X										
NavigationView											X										
Observation																				X	
ObservationController														X							
ObservationManager				X																	
ObservationView														X							
Practice																					X
PracticeController															X						
PracticeView															X						
ProcessArea	X																				
ProcessAreaController								X													
ProcessAreaView								X													

Tabla 7.4. Modelo de clases de diseño vs. Modelo de clases de análisis.

Modelo de despliegue vs Modelo de clases de diseño

Todas las clases y subsistemas de la aplicación se empaquetan dentro de un único componente. Por este motivo, no es necesario construir ninguna matriz que relacione clases con componentes de la aplicación.

7.8 Especificaciones de construcción del sistema

7.8.1 Especificación del entorno de construcción

El entorno necesario para la construcción de los componentes del sistema es el siguiente:

Hardware y software de base

- Computadora Personal Pentium II o superior, con al menos 64 MB de memoria RAM.
- Ambiente de ejecución y/o desarrollo de la plataforma J2SE, más conocido como JDK (Java Development Kit) o JRE (Java Runtime Environment), versión 1.4.2 o superior.

Herramientas de desarrollo

- Entorno integrado de desarrollo Netbeans 3.5 [J2SE, 2004], se utilizará en la construcción de las interfaces gráficas de usuario.
- Entorno integrado de desarrollo Eclipse 2.1.1 [Eclipse, 2004], se utilizará como herramienta principal para la codificación, compilación y depuración del sistema.
- Herramienta CASE Enterprise Architect 3.6 [EA, 2004], se utilizará para la generación inicial del código a partir de los modelos de diseño.

7.8.2 Descripción de subsistemas de construcción y dependencias

Los subsitemas de construcción se obtienen como una extensión directa a los subsistemas de diseño. El siguiente reporte de *Enterprise Architect* muestra el Modelo de componentes a utilizarse en la construcción del sistema. Cabe aclarar que en el mismo se ha optado por un nivel de granularidad grueso (nivel de paquete).

Modelo de componentes

En este modelo se muestran los componentes del sistema, contemplando las interfaces y las vinculaciones existentes entre ellos. El diagrama de la figura 7.42 muestra los componentes mencionados.

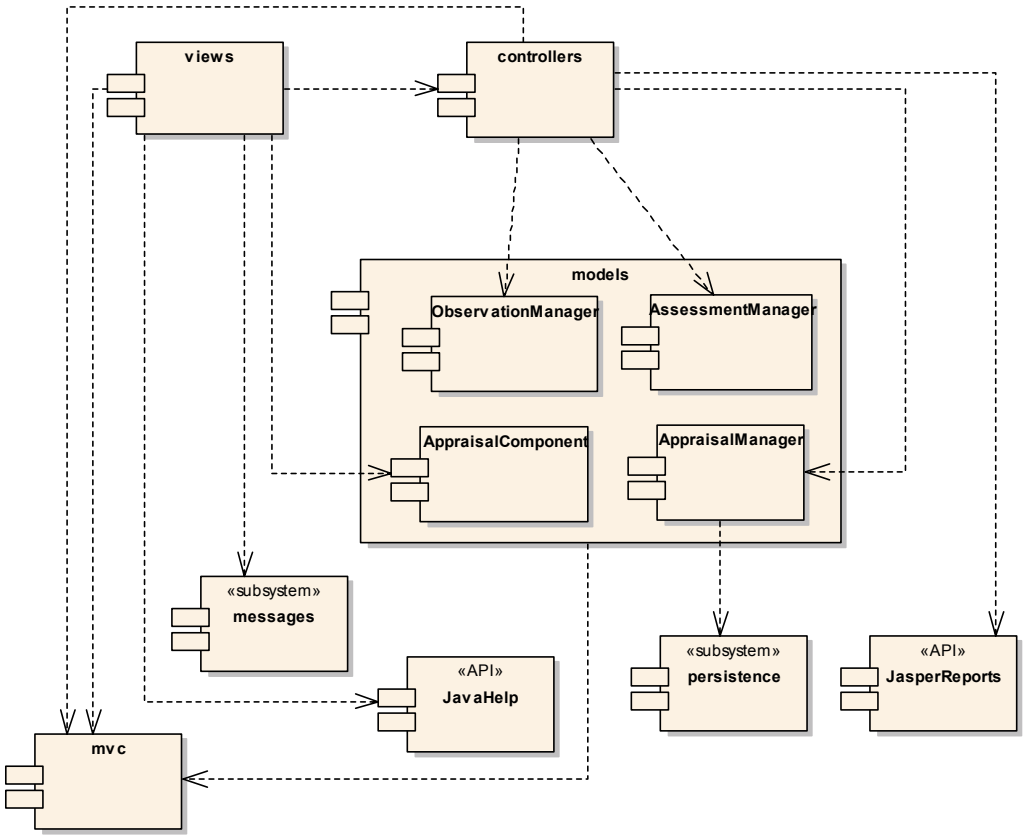

Figura 7.42. Componentes del sistema.

mvc

public Component: Este componente encapsula al framework "*Model View Controller*" que provee el mecanismo principal de funcionamiento de la aplicación.

views

public Component: Este componente encapsula a todas las interfaces de usuario.

controllers

public Component: Este componente encapsula a todos los controladores de interfaz de usuario, encargados de procesar las acciones que hace el usuario en los elementos de la interfaz gráfica.

models

public Component: Este componente encapsula el paquete "*models*" del diseño. Dentro del mismo se encuentran los siguientes componenetes:

- Controladores del negocio (clases *AppraisalManager*, *AssessmentManager* y *ObservationManager*), encargados de la manipulación de las entidades del negocio.
- Entidades del negocio (clases *Appraisal*, *MaturityLevel*, *ProcessArea*, *Goal*, *Practice* y *Observation*), encargadas de la representación del estado actual de la evaluación.

  Los controladores se encuentran representados con sus interfaces específicas, mientras que las entidades se han representado por su interfaz genérica (*AppraisalComponent*).

AppraisalComponent

public Component

Implements: Appraisal, AppraisalComponent, Goal, MaturityLevel, Observation, Practice, ProcessArea. : Este componente encapsula a todos los componentes del modelo (clases *Appraisal*, *MaturityLevel*, *ProcessArea*, *Goal*, *Practice* y *Observation*).

AppraisalManager

public Component

Implements: AppraisalFactory, AppraisalManager. : Este componente encapsula al controlador encargado de la administración de evaluaciones.

AssessmentManager

public Component

Implements: AssessmentManager. : Este componente encapsula al controlador encargado de la evaluación de las distintas partes del modelo.

ObservationManager

public Component

Implements: ObservationManager. : Este componente encapsula al controlador encargado de la administración de observaciones.

messages

public «subsystem» Component: Este componente encapsula al subsistema de mensajes. La interfaz que expone al exterior está dada por la clase *MessageManager*.

persistence

public «subsystem» Component: Este componente encapsula al subsistema de persistencia. La interfaz que expone al exterior está dada por la clase *PersistenceManager*.

JavaHelp

public «API» Component: Este componente representa al producto JavaHelp utilizado para el manejo de ayudas dentro de la aplicación.

JasperReports

public «API» Component: Este componente representa al producto JasperReports utilizado para la generación de reportes.

7.8.3 Plan de integración del sistema

Teniendo en cuenta los componentes expuestos en el apartado anterior, se define el siguiente plan para la integración de los mismos. El orden de construcción y prueba de los componentes será el que se expone en la tabla 7.5 a continuación.

Orden	*Componente*	*Descripción*
1	persistence	Construcción y prueba unitaria de la interfaz
2	messages	Construcción y prueba unitaria de la interfaz
3	mvc	Construcción del framework
4	Models->AppraisalComponent	Construcción y prueba de entidades del modelo (integrando las entidades entre sí)
5	models->AppraisalManager	Construcción y prueba de la interfaz (integrado con entidades y persistence)
6	models->AssessmentManager	Construcción y prueba de la intefaz (integrado con entidades)
7	models->ObservationManager	Construcción incluyendo y prueba de la interfaz (integrado con entidades)
8	controllers	Construcción y prueba de integración con los controladores de models
9	views	Construcción y prueba de integración con messages y entidades de models

Tabla 7.5. Orden para la construcción y prueba de los componentes.

8. PLAN DE PRUEBAS

En este capítulo se define el plan de pruebas. El objetivo de este plan es definir los niveles de pruebas que se utilizarán durante el desarrollo del proyecto, establecer las estrategias de trabajo, planificar las actividades a llevar a cabo, establecer los criterios de verificación y aceptación, y definir los productos resultantes de las mismas.

La definición del plan se inicia durante el proceso ASI, y su información se actualiza durante los procesos DSI, CSI e IAS contemplados en la metodología.

8.1 Especificación de los niveles de prueba

A continuación se tratan los distintos niveles de prueba que se llevaran a cabo durante el desarrollo de proyecto.

En todos los casos donde se menciona la palabra “defecto”, se entiende por tal a cualquier respuesta del sistema que se aparte de la esperada. Los defectos se califican empleando tres niveles de severidad: Alta, Media y Baja.

8.1.1 Pruebas unitarias y de integración

Las pruebas unitarias tienen como objetivo verificar la funcionalidad y estructura de cada uno de los componentes de manera individual. Estas pruebas a menudo requieren la creación de componentes auxiliares que simulen la funcionalidad de otros componentes que interactúan con el que se está probando.

Las pruebas de integración tienen como objetivo verificar el correcto ensamblaje entre los distintos componentes del sistema, comprobando que los mismos funcionan correctamente a través de sus interfaces, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales.

De manera de evitar la generación de componentes auxiliares, las pruebas unitarias se fusionarán con las pruebas de integración siguiendo una estrategia de integración incremental.

Perfiles involucrados

Tesista: Responsable del diseño, la ejecución y la evaluación de las pruebas.
Director: Responsable de controlar que las pruebas se efectúen correctamente.

Planificación temporal

Las pruebas se diseñan durante el proceso DSI y se ejecutan durante el proceso CSI. La planificación de las actividades de diseño, ejecución y evaluación de las pruebas se han incluido dentro del capítulo **5. Plan general del proyecto** (apartado **5.1.2 Planificación**).

Criterios de verificación y aceptación

Las pruebas se dan por concluidas exitosamente cuando todos los componentes seleccionados para las mismas hayan sido probados y no se detectan más defectos.

Generación y mantenimiento de verificaciones y casos de prueba

Las verificaciones y casos de pruebas serán automatizados por medio de la herramienta JUnit [JUnit, 2004]. Para cada componente se definirá un conjunto de casos de prueba con sus respectivas verificaciones sobre los resultados obtenidos, todos ellos documentados y automatizados en JUnit. El enfoque que se utilizará en estas pruebas es el de caja negra.

Análisis y evaluación de los resultados

La ejecución de los casos de prueba en JUnit genera excepciones solamente en el caso de fallas. Las excepciones obtenidas durante la ejecución de las pruebas serán analizadas y los defectos que las mismas representan serán corregidos durante el desarrollo.

Entregables resultantes de las pruebas

Dado que tanto las pruebas unitarias y de integración como la corrección de los defectos encontrados en las mismas se efectúan durante el desarrollo, no se generan entregables donde se documente el resultado de estas pruebas.

Los únicos entregables resultantes de las pruebas son el código fuente de las mismas (casos de prueba JUnit), y el código fuente de la aplicación probado, el cual pasa sin fallas por todos los casos de prueba unitarios y de integración.

8.1.2 Pruebas del sistema

Las pruebas del sistema tienen como objetivo encontrar defectos en el funcionamiento del sistema completo. Dentro de estas pruebas se incluyen las siguientes:

- Pruebas funcionales: dirigidas a asegurar que el sistema realiza correctamente todas las funciones detalladas en los requerimientos.
- Pruebas de rendimiento: orientadas a determinar si los tiempos de respuesta se encuentran dentro de los límites establecidos en las especificaciones del sistema.
- Pruebas de facilidad de uso: comprueban la experiencia del usuario en el uso del sistema, asegurando que el mismo se acomode al modo de trabajo de los usuarios, y que le aporta elementos que faciliten el ingreso y la recuperación de los datos.

NOTA: dado que la instalación del sistema consiste en copiar un conjunto de archivos en un directorio, no se considerarán pruebas de instalación. Por otra parte, dado que el sistema está implementado sobre la plataforma Java, la cual es suficientemente madura y comprobadamente portable entre sistemas operativos, no se realizarán pruebas de portabilidad.

Perfiles involucrados

Tesista: Responsable del diseño de todas las pruebas, y de la ejecución y evaluación de las pruebas funcionales y de rendimiento.
Usuario: Responsable de la ejecución y evaluación de las pruebas de facilidad de uso.
Director: Responsable de controlar que las pruebas se efectúen correctamente.

Planificación temporal

Las pruebas se diseñan durante los procesos ASI y DSI, y se ejecutan durante el proceso CSI. La planificación de las actividades de diseño, ejecución y evaluación de

las pruebas se han incluido dentro del capítulo **5. Plan general del proyecto** (apartado **5.1.2 Planificación**).

Criterios de verificación y aceptación

Las pruebas se darán por concluidas exitosamente cuando todos los casos de prueba hayan sido ejecutados y no se detecten más defectos de severidad Media o Alta. Podrán existir defectos de severidad Baja, siempre y cuando no superen la cantidad de 5.

Generación y mantenimiento de verificaciones y casos de prueba

Las verificaciones y casos de prueba se definen en la sección **7.9.3 Especificación técnica de niveles de prueba** de este mismo documento. Sin embargo, para cada tipo de prueba se utilizará una aproximación diferente.

- Pruebas funcionales: para cada caso de uso del sistema, existirá un apartado con los casos de prueba funcionales.
- Pruebas de rendimiento: existirá un único apartado con los casos de prueba de rendimiento.
- Pruebas de facilidad de uso: no se definirán casos de prueba formales para este tipo de pruebas. Se dejará a cargo del usuario la utilización del sistema y la evaluación de la usabilidad del mismo.

Análisis y evaluación de los resultados

Se analizarán los resultados de la ejecución de los casos de prueba en busca de defectos o incidentes que requieran correcciones.

Entregables resultantes de las pruebas

Se generará un documento con los resultados de la ejecución de los casos de prueba.

8.1.3 Pruebas de implantación y aceptación

El objetivo de las pruebas de implantación es comprobar el correcto funcionamiento del sistema en su entorno operativo. Estas pruebas permiten que el usuario realice la aceptación del sistema desde el punto de vista operativo, lo que incluye la instalación y el mantenimiento del sistema.

El objetivo de las pruebas de aceptación es validar que el sistema cumple con el funcionamiento esperado y permitir al usuario que determine su aceptación desde el punto de vista de la funcionalidad y el rendimiento.

Dentro de las pruebas previstas para el proyecto, se fusionarán las pruebas de implantación con las de aceptación.

NOTA: dadas las características de la herramienta resultante del proyecto (herramienta standalone que no posee requerimientos de instalación más allá de la plataforma Java), las pruebas a considerar en este nivel abarcan solamente las de aceptación por parte del usuario.

Perfiles involucrados

Tesista: Responsable de asistir al usuario en el diseño las pruebas, la ejecución y la evaluación de las mismas.
Usuario: Responsable del diseño, la ejecución y la evaluación de las pruebas, asistido por el tesista.
Director: Responsable de controlar que las pruebas se efectúen correctamente.

Planificación temporal

Las pruebas se diseñan durante los procesos ASI y DSI, y se ejecutan durante el proceso IAS. La planificación de las actividades de diseño, ejecución y evaluación de las pruebas se han incluido dentro del capítulo **5. Plan general del proyecto** (apartado **5.1.2 Planificación**).

Criterios de verificación y aceptación

- Las pruebas se darán por concluidas exitosamente cuando el usuario confirme su aceptación del sistema. El criterio de aceptación del usuario es el siguiente: no debe existir ningún defecto con severidad Media o Alta, pueden existir hasta 5 defectos con severidad Baja.

Generación y mantenimiento de verificaciones y casos de prueba

Se utilizarán los casos de prueba de funcionalidad y rendimiento generados para las pruebas del sistema.

Análisis y evaluación de los resultados

Se analizarán los resultados de la ejecución de los casos de prueba en busca de defectos o incidentes que requieran correcciones.

Entregables resultantes de las pruebas

El único entregable resultante de estas pruebas será el acuerdo de aceptación por parte del usuario, donde el mismo manifieste su aceptación del sistema terminado.

8.1.4 Pruebas de regresión

El objetivo de estas pruebas es comprobar que los cambios sobre un componente del sistema no introducen errores en otros componentes.

Las pruebas de regresión se llevarán a cabo cada vez que se haga un cambio sobre un componente, ya sea para corregir un error como para realizar una mejora en el mismo. Dichas pruebas consistirán en repetir las pruebas unitarias y de integración, y las pruebas del sistema.

8.2 Especificación del entorno de pruebas

Para la realización de las pruebas se contará con dos entornos separados:

- Entorno de desarrollo: será utilizado para las pruebas unitarias y de integración.
- Entorno de pruebas: será utilizado para las pruebas de sistema, y las de implantación y aceptación.

A continuación se enuncian los requisitos para ambos entornos.

Requisitos de hardware y software

Los requisitos son básicamente los mismos que se necesitan para la ejecución del sistema.

- Computadora Personal Pentium II o superior, con al menos 64 MB de memoria RAM.
- Ambiente de ejecución y/o desarrollo de la plataforma J2SE, más conocido como JDK (Java Development Kit) o JRE (Java Runtime Environment), versión 1.4.2 o superior.

Herramientas auxiliares

- Entorno integrado de desarrollo Eclipse [Eclipse, 2004].
- Librerías de ejecución de la herramienta JUnit [JUnit, 2004].

8.3 Especificación técnica de niveles de prueba

En esta sección se vuelca el diseño de los casos de prueba y los procedimientos de prueba para cada uno de los niveles definidos en el apartado 7.9.1.

8.3.1 Pruebas unitarias y de integración

Para el diseño de estas pruebas se toma como base el Modelo de componentes y el Plan de integración planteados en los apartados 7.8.2 y 7.8.3.

Diseño de casos de prueba

A continuación se muestran los casos de prueba diseñados para cada uno de los componentes. En el caso de los componentes de nivel inferior (*persistence*, *messages* y *AppraisalComponent*) las pruebas son netamente unitarias. En el caso de los componentes de nivel superior (*AppraisalManager*, *AssessmentManager*, *ObservationManager*, *controllers* y *views*), las pruebas son netamente de integración.

Para el caso de los subsistemas *persistence* y *messages*, y para los componentes del paquete models (*AppraisalManager, AssessmentManager, ObservationManager, AppraisalComponent* y sus clases derivadas), los casos de prueba se programan en Java utilizando la librería JUnit. Esto hace que los mismos sean automatizados y repetibles, lo cual brinda a su vez la posibilidad de hacer pruebas de regresión automáticas en los niveles de prueba unitaria y de integración.

En el caso de los *controllers* y las *views*, las pruebas de integración son manuales y no repetibles, ya que consisten en "conectar" las interfaces de usuario con la lógica de negocios de la aplicación (concentrada en el paquete "*models*").

Abreviaturas:

CP: Caso de prueba
da: *Direct artifacts* (Artefactos Directos)
ia: *Indirect artifacts* (Artefactos Indirectos)
obs: *Observations* (Observaciones)
oos: *Out of scope* (Fuera de alcance)
ne: *No evidence* (Sin evidencia)

NOTA: los nombres asignados a las variables o valores utilizados en los casos de prueba se corresponden directamente con los utilizados en el código fuente de la aplicación. Por tal motivo, todos ellos se encuentran en inglés (tal como están en el código).

CP	***Componente***	***Descripción***	***Precondiciones***	***Entradas***	***Salidas***	***Postcondiciones***
1	Subsistema *persistence*	Prueba método *write* del *PersistenceManager*. Genera un objeto X de prueba y lo almacena mediante el *PersistenceManager*.	Crear un objeto (Objeto X) de prueba que tenga como atributos un String, un entero, una lista y una tabla	**object**: Objeto X de prueba **file**: TestFile.xml	Archivo TestFile.xml con la representación del Objeto X en XML	El archivo TestFile.xml queda almacenado en el disco rígido
2	Subsistema *persistence*	Prueba método *read* del *PersistenceManager*. Genera un objeto X de prueba y lo compara con otro leído de archivo por medio del *PersistenceManager*	**1.** Crear un objeto (Objeto X) de prueba que tenga como atributos un String, un entero, una lista y una tabla **2.** Ejecutar el CP 1 de manera que exista el archivo TestFile.xml con la representación en XML del Objeto X	**file**: TestFile.xml	El objeto leído debe ser igual al creado	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	Subsistema *messages*	Prueba método *getMessage* del *MessageManager*. Inicializa el *MessageManager* para un idioma determinado (español) y recupera un mensaje de prueba con un argumento asociado	Debe existir el archivo messages_es.properties en el *classpath* (camino de acceso a las clases java) de la aplicación. El archivo debe contener la sig. entrada: TEST1 = Mensaje con argumento {0}	**key**: TEST1 **params**: array de objetos con un único elemento llamado "Argumento 1"	El mensaje recuperado debe ser "Mensaje con argumento Argumento 1"	Ninguna
4	Clase *Appraisal*	Prueba método *setSaved*.	**1.** Crear una evaluación y un *listener* (objeto que escucha los cambios en otro objeto) interesado en los cambios de la misma. **2.** Agregar el *listener* a la evaluación.	**newVal**: true	isSaved: trae Eventos recibidos en el listener: "Welcome", "Appraisal Saved"	Ninguna
5	Clase *Appraisal*	Prueba método *addObservation*.	**1.** Crear una evaluación y un *listener* interesado en los cambios de la misma. **2.** Agregar el *listener* a la evaluación. **3.** Crear una observación (newObs)	**obs**: newObs	getObservation(1): newObs Eventos recibidos en el listener: "Welcome", "New Observation"	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
6	Clase *Appraisal*	Prueba método *updateObservation*.	**1.** Crear una evaluación y un *listener* interesado en los cambios de la misma. **2.** Agregar el *listener* a la evaluación. **3.** Crear una observación (newObs) y agregarla a la evaluación. **4.** A continuación modificar la descripción de la observación (newObs)	**obs**: newObs	getObservation(1): newObs Eventos recibidos en el listener: "Welcome", "Observation Changed"	Ninguna
7	Clase *Appraisal*	Prueba método *removeObservation*.	**1.** Crear una evaluación y un *listener* interesado en los cambios de la misma. **2.** Agregar el *listener* a la evaluación. **3.** Crear una observación (newObs) y agregarla a la evaluación.	**obs**: newObs	getObservation(1): null Eventos recibidos en el listener: "Welcome", "Observation Deleted"	Ninguna
8	Clase *Appraisal*	Prueba método *setDirty*.	**1.** Crear una evaluación y un *listener* interesado en los cambios de la misma. **2.** Agregar el *listener* a la evaluación.	**newVal**: true	Evaluación: dirty: true Eventos recibidos en el listener: "Welcome", "Suggestion Changed"	Ninguna
9	Clase *Appraisal*	Prueba método *addChild*. Setea un nivel de madurez como hijo de una evaluación. Luego pregunta a la evaluación por el hijo recién agregado.	**1.** Crear una evaluación y un nivel de madurez. **2.** Crear un *listener* interesado en los cambios de la misma. **3.** Agregar el *listener* a la evaluación.	En la evaluación (Appraisal): **name**: "L2" **child**: MaturityLevel	getChild("L2") igual al MaturityLevel Eventos recibidos en el listener: "Welcome", "New Child"	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
10.1	Clase *MaturityLevel*	Prueba método *regenerateSuggestion*. Setea un par de áreas de proceso cualquiera como hijas de un nivel de madurez cualquiera y varía la valoración elegida de cada una de ellas, de manera de abarcar todas las reglas de inferencia.	**1.** Crear un nivel de madurez y dos áreas de proceso. **2.** Agregar las áreas de proceso como hijas del nivel de madurez. **3.** Crear un *listener* interesado en los cambios del nivel de madurez. **4.** Agregar el listener al nivel de madurez.	(Subcaso 1) **val pa1**: SATISFIED **val pa2**: NOT_APPLICABLE	(Subcaso 1) valoración: REACHED dirty: true Eventos recibidos en el listener: “Welcome”, “Suggestion Changed”	Ninguna
10.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **val pa1**: UNSATISFIED **val pa2**: NOT_APPLICABLE	(Subcaso 2) valoración: NOT_REACHED dirty: true	Ídem anterior
10.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **val pa1**: NOT_RATED **val pa2**: NOT_APPLICABLE	(Subcaso 3) valoración: NOT_REACHED dirty: true	Ídem anterior
10.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **val pa1**: null **val pa2**: NOT_APPLICABLE	(Subcaso 4) valoración: null dirty: true	Ídem anterior
11	Clase *MaturityLevel*	Prueba método *setDirty*. Setea un nivel de madurez como hijo de una evaluación, e invoca a setDirty para vericar que la notificación se propague al padre.	**1.** Crear una evaluación y un nivel de madurez. **2.** Agregar el nivel de madurez como hijo de la evaluación. **3.** Crea un *listener* interesado en los cambios del nivel de madurez. **4.** Agrega el *listener* al nivel de madurez.	**newVal**: true	Nivel de madurez: dirty: true Evaluación: dirty: trae Eventos recibidos en el listener: “Welcome”, “Suggestion Changed”	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
12	Clase *MaturityLevel*	Prueba método *addChild*. Setea un área de proceso como hija de un nivel de madurez. Luego pregunta al nivel de madurez por el hijo recién agregado.	**1.** Crear un nivel de madurez y un área de proceso. **2.** Crear un *listener* interesado en los cambios del nivel de madurez. **3.** Agrega el *listener* al nivel de madurez.	En el MaturityLevel: **name**: “L2.PA1” **child**: ProcessArea	getChild(“L2.PA1”) igual a la ProcessArea Eventos recibidos en el listener: “Welcome”, “New Child”	Ninguna
13.1	Clase *ProcessArea*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor del área de proceso cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	**1.** Crear un nivel de madurez y un área de proceso. **2.** Agregar el área de proceso como hija del nivel de madurez.	(Subcaso 1) **new_assessment**: SATISFIED	Valoración sugerida para el nivel de madurez: (Subcaso 1) REACHED	Ninguna
13.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **new_assessment**: NOT_APPLICABLE	(Subcaso 2) REACHED	Ídem anterior
13.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **new_assessment**: UNSATISFIED	(Subcaso 3) NOT_REACHED	Ídem anterior
13.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **new_assessment**: NOT_RATED	(Subcaso 4) NOT_REACHED	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
14.1	Clase *ProcessArea*	Prueba método *regenerateSuggestion*. Setea un par de objetivos como hijos del área de proceso y varía la valoración elegida de cada una de ellos, de manera de barrer todas las reglas de inferencia.	**1.** Crear un área de proceso y dos objetivos. **2.** Agregar los objetivos como hijos del área de proceso. **3.** Crear un *listener* interesado en los cambios del área de proceso. **4.** Agrega el *listener* al área de proceso.	(Subcaso 1) **oos**: true	(Subcaso 1) valoración: NOT_APPLICABLE dirty: true Eventos recibidos en el listener: "Welcome", "Suggestion Changed"	Ninguna
14.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **oos**: false **ne**: true	(Subcaso 2) valoración: NOT_RATED dirty: true	Ídem anterior
14.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **val goal1**: SATISFIED **val goal2**: SATISFIED	(Subcaso 3) valoración: SATISFIED dirty: true	Ídem anterior
14.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **val goal1**: UNSATISFIED **val goal2**: SATISFIED	(Subcaso 4) valoración: UNSATISFIED dirty: true	Ídem anterior
14.5	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 5) **val goal1**: null **val goal2**: SATISFIED	(Subcaso 4) valoración: null dirty: true	Ídem anterior
15.1	Clase *ProcessArea*	Prueba método *generateSuggestion*. Invoca al método con distintos valores de Fuera de alcance y Sin evidencia, de manera de ejercitar todas las reglas de inferencia.	**1.** Crear un listener interesado en los cambios del área de proceso. **2.** Agregar el listener al área de proceso.	(Subcaso 1) **oos**: true **ne**: false	Valoración sugerida: (Subcaso 1) NOT_APPLICABLE Eventos recibidos en el listener: "Welcome", "Suggestion Generated"	Ninguna
15.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **oos**: false **ne**: true	(Subcaso 2) NOT_RATED	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
16.1	Clase *Goal*	Prueba método *regenerateSuggestion*. Setea un par de prácticas como hijas del objetivo y varía la valoración elegida y las observaciones de cada una de ellas, de manera de barrer todas las reglas de inferencia.	**1.** Crear un objetivo y dos prácticas. **2.** Agregar las prácticas como hijas del objetivo. **3.** Crear un listener interesado en los cambios del objetivo. **4.** Agregar el listener al objetivo.	(Subcaso 1) **val p1**: FULLY_IMPLEMENTED **obs p1**: WEAKNESS, NOT_SIGNIFICANT **val p2**: LARGELY_IMPLEMENTED **obs p2**: WEAKNESS, NOT_SIGNIFICANT	(Subcaso 1) valoración: SATISFIED dirty: true Eventos recibidos en el listener: “Welcome”, “Suggestion Changed”	Ninguna
16.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **val p1**: FULLY_IMPLEMENTED **obs p1**: WEAKNESS, SIGNIFICANT **val p2**: LARGELY_IMPLEMENTED **obs p2**: WEAKNESS, SIGNIFICANT	(Subcaso 2) valoración: UNSATISFIED dirty: true	Ídem anterior
16.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **val p1**: PARTIALLY_IMPLEMENTED **obs p1**: ninguna **val p2**: NOT_IMPLEMENTED **obs p2**: ninguna	(Subcaso 3) valoración: UNSATISFIED dirty: true	Ídem anterior
16.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **val p1**: null **obs p1**: ninguna **val p2**: FULLY_IMPLEMENTED **obs p2**: ninguna	(Subcaso 4) valoración: null dirty: true	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
17.1	Clase *Goal*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor del objetivo cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	**1.** Crear un área de proceso y un objetivo. **2.** Agregar el objetivo como hijo del área de proceso.	(Subcaso 1) **new_assessment**: SATISFIED	Valoración sugerida para el área de proceso: (Subcaso 1) SATISFIED	Ninguna
17.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **new_assessment**: UNSATISFIED	(Subcaso 2) UNSATISFIED	Ídem anterior
18.1	Clase *Practice*	Prueba método *regenerateSuggestion* para el caso de práctica a nivel de instancia. Setea valores para Artefactos directos, Artefactos indirectos y observaciones de manera de ejercitar todas las reglas de inferencia.	**1.** Crear una práctica y un listener interesado en los cambios en la misma. **2.** Agregar el listener a la práctica.	Los subcasos se ejecutan en orden. (Subcaso 1) **da**: APPROPIATE **ia**: CONFIRM **obs**: tipo STRENGTH, impacto NOT_SIGNIFICANT	Valoración sugerida: (Subcaso 1) FULLY_IMPLEMENTED dirty: true Eventos recibidos en el listener: “Welcome” y 4 “Suggestion Changed”	Ninguna
18.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **obs**: tipo WEAKNESS, impacto SIGNIFICANT	(Subcaso 2) LARGELY_IMPLEMENTED dirty: true	Ídem anterior
18.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **da**: NOT_APPROPIATE **ia**: SUGGEST	(Subcaso 3) PARTIALLY_IMPLEMENTED dirty: true	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
18.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **da**: UNKNOWN **ia**: UNKNOWN	(Subcaso 4) NOT_IMPLEMENTED dirty: true	Ídem anterior
19.1	Clase *Practice*	Prueba método *regenerateSuggestion* para el caso de práctica a nivel de conjunto de instancias. Asigna dos instancias como hijas de una práctica, y setea diferentes valores de valoración a cada una de las instancias para ejercitar todas las reglas de inferencia.	Ninguna	Los subcasos se ejecutan en orden. (Subcaso 1) **inst1**: FULLY_IMPLEMENTED **inst2**: FULLY_IMPLEMENTED	Valoración sugerida: (Subcaso 1) FULLY_IMPLEMENTED dirty: true	Ninguna
19.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **inst1**: NOT_IMPLEMENTED **inst2**: NOT_IMPLEMENTED	(Subcaso 2) NOT_IMPLEMENTED dirty: true	Ídem anterior
19.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **inst1**: FULLY_IMPLEMENTED **inst2**: LARGELY_IMPLEMENTED	(Subcaso 3) LARGELY_IMPLEMENTED dirty: true	Ídem anterior
19.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **inst1**: PARTIALLY_IMPLEMENTED **inst2**: LARGELY_IMPLEMENTED	(Subcaso 4) LARGELY_PARTIALLY_IMPLEMENTED dirty: true	Ídem anterior
19.5	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 5) **inst1**: NOT_IMPLEMENTED **inst2**: LARGELY_IMPLEMENTED	(Subcaso 5) NOT_PARTIALLY_LARGELLY_IMPLEMENTED dirty: true	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
20	Clase *Practice*	Prueba método *removeObservation*. Setea valores para Artefactos directos, Artefactos indirectos y observaciones de manera de obtener una valoración sugerida. A continuación elimina una observación provocando un cambio en la valoración sugerida.	Setear los siguientes valores en la práctica: **obs1**: tipo STRENGTH impacto NOT_SIGNIFICANT **obs2**: tipo WEAKNESS impacto SIGNIFICANT **da**: APPROPIATE **ia**: CONFIRM Con estos valores se tiene como valoración sugerida LARGELY_IMPLEMENTED	Argumento para removeObservation: **obs2**.	Valoración sugerida: FULLY_IMPLEMENTED	Ninguna
21.1	Clase *Practice*	Prueba método *generateSuggestion*. Invoca al método con distintos valores de Artefactos directos, Artefactos indirectos y Observaciones de manera de ejercitar todas las reglas de inferencia.	Ninguna	Los subcasos se ejecutan uno a continuación del otro. (Subcaso 1) **da**: APPROPIATE **ia**: CONFIRM **obs**: una con tipo STRENGTH e impacto NOT_SIGNIFICANT	Valoración sugerida: (Subcaso 1) FULLY_IMPLEMENTED	Ninguna
21.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **da**: APPROPIATE **ia**: CONFIRM **obs**: una con tipo WEAKNESS e impacto SIGNIFICANT	(Subcaso 2) LARGELY_IMPLEMENTED	Ídem anterior
21.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **da**: NOT_APPROPIATE **ia**: SUGGEST **obs**: una con tipo WEAKNESS e impacto SIGNIFICANT	(Subcaso 3) PARTIALLY_IMPLEMENTED	Ídem anterior

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
21.4	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 4) **da**: UNKNOWN **ia**: UNKNOWN **obs**: una con tipo WEAKNESS e impacto SIGNIFICANT	(Subcaso 4) NOT_IMPLEMENTED	Ídem anterior
22.1	Clase *Practice*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor de la práctica cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	**1.** Crear un objetivo y una práctica. **2.** Agregar la práctica como hija del objetivo.	(Subcaso 1) **new_assessment**: LARGELY_IMPLEMENTED	Valoración sugerida para el objetivo: (Subcaso 1) SATISFIED	Ninguna
22.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **new_assessment**: PARTIALLY_IMPLEMENTED	(Subcaso 2) UNSATISFIED	Ídem anterior
23	Clase *Observation*	Prueba método *update*. Crea una práctica y un objetivo. Asigna el objetivo a la práctica. Luego modifica el objetivo para verificar que se regenere la valoración sugerida en la práctica.	**1.** Crear una práctica y una observación. **2.** Agregar la observación a la práctica. Datos iniciales de la observación: **type**: STRENGTH **impact**: SIGNIFICANT Valoración sugerida en la práctica: FULLY_IMPLEMENTED	**type**: WEAKNESS **impact**: SIGNIFICANT	Valoración sugerida en la práctica: LARGELY_IMPLEMENTED	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
24	Clase *Observation*	Prueba método *notifyDelete*. Crea una práctica y un objetivo. Asigna el objetivo a la práctica. Luego invoca a *notifyDelete* en el objetivo para verificar que se regenere la valoración sugerida en la práctica.	**1.** Crear una práctica y una observación. **2.** Agregar la observación a la práctica. Datos iniciales de la observación: **type**: WEAKNESS **impact**: SIGNIFICANT Valoración sugerida en la práctica: LARGELY_IMPLEMENTED		Valoración sugerida en la práctica: FULLY_IMPLEMENTED	Ninguna
25	Clase *AppraisalManager*	Prueba método *createAppraisal*. Crea una evaluación y verifica si se ha creado correctamente.	Crear un conjunto de instancias para agregar a la evaluación (list)	**target**: “Test organization” **scope**: PRACTICES **instances**: list	Sobre el objeto Appraisal: getTarget(): “Test organization” getScope(): PRACTICES getInstances(): list getChildren(): objetos MaturityLevel “L2”, “L3”, “L4”, “L5”	Ninguna
26.1	Clase *AppraisalManager*	Prueba método *initLevel*. Crea una evaluación, inicializa un nivel de madurez dentro de la misma y verifica si se ha inicializado correctamente.	Ninguna	(Subcaso 1) **scope**: PRACTICES **level**: L2	(Subcaso 1) MaturityLevel L2 con 7 ProcessAreas. PA1 con un Objetivo genérico y uno específico. Objetivo específico con 4 prácticas específicas.	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
26.2	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 2) **scope**: GOALS **level**: L2	(Subcaso 2) MaturityLevel L2 con 7 ProcessAreas. PA1 con un Objetivo genérico y uno específico. Objetivo específico sin prácticas específicas.	Ídem anterior
26.3	Ídem anterior	Ídem anterior	Ídem anterior	(Subcaso 3) **scope**: PROCESSAREAS **level**: L2	(Subcaso 3) MaturityLevel L2 con 7 ProcessAreas. PA1 sin Objetivos genéricos ni específicos.	Ídem anterior
27	Clase *AppraisalManager*	Prueba método *saveAppraisal*. Crea una evaluación, la completa la almacena.	Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal)	**appraisal**: newAppraisal **file**: AppraisalTest.xml	Ninguna	El archivo AppraisalTest.xml contiene la representación de la evaluación completa en XML
28	Clase *AppraisalManager*	Prueba método *openAppraisal*. Crea una evaluación y la completa. Lee una evaluación de archivo y la compara con la anterior.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Leer de archivo una evaluación igual a la anterior y compara ambas.	**file**: AppraisalTest.xml	Evaluación creada igual a la evaluación leída	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
29	Clase *AssessmentManager*	Prueba método *setAssessment* para áreas de proceso. Crea una evaluación con un nivel de madurez y sus áreas de proceso. Setea las valoraciones de todas las áreas de proceso y verifica que se genere automáticamente la valoración del nivel de madurez.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Obtener las 7 áreas de proceso (pa1 a pa7) del nivel L2.	Un subcaso para cada pa (1 a 7) **process_area**: pa1 a pa7 **oos**: false **ne**: false **suggestion**: SATISFIED **comment**: “None”	(Subcasos 1 al 6) Valoración del Nivel L2: null (Subcaso 7) Valoración del Nivel L2: REACHED	Ninguna
30	Clase *AssessmentManager*	Prueba método *setAssessment* para objetivos. Crea una evaluación con un nivel de madurez, sus áreas de proceso y objetivos. Setea las valoraciones de todos los objetivos de un área de proceso y verifica que se genere automáticamente la valoración del área de proceso.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Obtener los 2 objetivos (g1, g2) del área de proceso 1 correspondiente al nivel 2 (L2.PA01).	Un subcaso para cada objetivo **goal**: g1, g2 **sugg**: SATISFIED **assess**: SATISFIED **comment**: “None”	(Subcaso 1) Valoración del área L2.PA01: null (Subcaso 2) Valoración del área L2.PA01: SATISFIED	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
31	Clase *AssessmentManager*	Prueba método *setAssessment* para prácticas. Crea una evaluación con un nivel de madurez, sus áreas de proceso, objetivos, y prácticas. Setea las valoraciones de todas las prácticas de un objetivo y verifica que se genere automáticamente la valoración del objetivo.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Obtener las 5 prácticas (p1 a p5) del objetivo 1 correspondiente al área 1 del nivel 2 (L2.PA01.SG1).	Un subcaso para cada práctica (p1 a p5) **practice**: p1 a p5 **da**: APPROPIATED **ia**: CONFIRM **obs**: appraisal.getObservations() **sugg**: FULLY_IMPLEMENTED **assess**: LARGELY_IMPLEMENTED **comment**: “None”	(Subcasos 1 al 4) Valoración del objetivo L2.PA01.SG1: null (Subcaso 5) Valoración del objetivo L2.PA01.SG1: SATISFIED	Ninguna
32	Clase *ObservationManager*	Prueba método *createObservation*. Crea una evaluación completa (incluyendo observaciones). Crea una nueva observación en la evaluación y verifica que se haya creado bien.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregar 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Crear una observación en forma manual (con nro identificatorio 4, testObs).	**appraisal**: newAppraisal **type**: WEAKNESS **impact**: SIGNIFICANT **description**: “None”	La observación nro 4 de la evaluación es igual a la creada manualmente (testObs)	Ninguna
33	Clase *ObservationManager*	Prueba método *updateObservation*. Crea una evaluación completa (incluyendo observaciones). Modifica una observación en la evaluación y verifica que se haya modificado bien.	**1.** Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal) **2.** Crear una observación en forma manual (con nro identificatorio 2, testObs).	**appraisal**: newAppraisal **number**: 2 **type**: WEAKNESS **impact**: SIGNIFICANT **description**: “None”	La observación nro 2 de la evaluación es igual a la creada manualmente (testObs)	Ninguna

CP	*Componente*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
34	Clase *ObservationManager*	Prueba método *deleteObservation*. Crea una evaluación completa (incluyendo observaciones). Elimina una observación en la evaluación y verifica que se haya eliminado bien.	Crear una evaluación con alcance a nivel de prácticas, agregarle 3 observaciones, e inicializar todos los niveles de madurez de la misma (newAppraisal)	**appraisal**: newAppraisal **number**: 2	La observación nro 2 de la evaluación es igual a null	Ninguna

Automatización de las pruebas

A continuación se muestra la forma en que los casos de prueba son automatizados dentro del entorno de desarrollo con la librería JUnit.

Para cada componente o clase a testear se crea un *TestCase* de JUnit. El *TestCase* contiene todos los casos de prueba asociados al objeto que se desea probar. La figura 8.1 muestra un ejemplo de automatización.

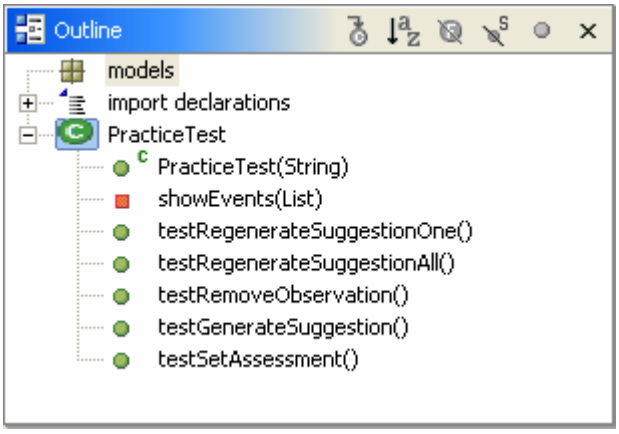

Figura 8.1. TestCase JUnit con todos los casos de prueba de un componente.

Cada uno de los tests es codificado en Java. La figura 8.2 muestra un ejemplo.

```
public void testSetAssessment() {
      Goal goal = new Goal("L2.PA1.G1");
      Practice p1 = new Practice("L2.PA1.G1.P1");
      TestListener listener = new TestListener();

      goal.addChild("L2.PA1.G1.P1", p1);
      p1.addListener(listener);

      p1.setAssessment(Practice.LARGELY_IMPLEMENTED);
      assertTrue(goal.getSuggestion().equals(Goal.SATISFIED));

      p1.setAssessment(Practice.PARTIALLY_IMPLEMENTED);
      assertTrue(goal.getSuggestion().equals(Goal.UNSATISFIED));

      Observation obs1 = new Observation(new Integer(1),
            Observation.WEAKNESS, Observation.SIGNIFICANT,
            "Test observation 1");
      p1.addObservation(obs1);
      p1.setAssessment(Practice.FULLY_IMPLEMENTED);
      assertTrue(goal.getSuggestion().equals(Goal.UNSATISFIED));

      // Chequeo de notificaciones
```

```
        List events = listener.getEvents();
        //showEvents(events);

    }
```

Figura 8.2. Código fuente de un TestCase JUnit.

Luego de definir todos los *TestCases* JUnit, se crea una *TestSuite* que contenga todos los *TestCases*, de manera de poder ejecutarlos a todos juntos.

La figura 8.3 muestra la *TestSuite* creada para los *TestCases* definidos.

```
public class AllTests {

        public static Test suite() {
            TestSuite suite = new TestSuite("Test for
 messages");
            //$JUnit-BEGIN$
            suite.addTest(new
 TestSuite(PersistenceManagerTest.class));
            suite.addTest(new
 TestSuite(MessageManagerTest.class));
            suite.addTest(new TestSuite(AppraisalTest.class));
            suite.addTest(new
 TestSuite(MaturityLevelTest.class));
            suite.addTest(new TestSuite(ProcessAreaTest.class));
            suite.addTest(new TestSuite(GoalTest.class));
            suite.addTest(new TestSuite(PracticeTest.class));
            suite.addTest(new TestSuite(ObservationTest.class));
            suite.addTest(new
 TestSuite(AppraisalManagerTest.class));
            suite.addTest(new
 TestSuite(AssessmentManagerTest.class));
            suite.addTest(new
 TestSuite(ObservationManagerTest.class));
            //$JUnit-END$
            return suite;
        }
}
```

Figura 8.3. Código fuente de una TestSuite JUnit.

Finalmente, la TestSuite puede ejecutarse dentro del entorno de desarrollo, reportando el éxito o fracaso de los casos de prueba.

La figura 8.4 muestra la ventana de ejecución de pruebas JUnit dentro del entorno integrado de desarrollo Eclipse, para un caso en el que deliberadamente se hace fallar un caso de prueba. Como puede verse en la figura, la herramienta reporta cuál es el caso de prueba fallido y en qué lugar del código de pruebas se produce la falla, lo que facilita enormemente la corrección de los defectos.

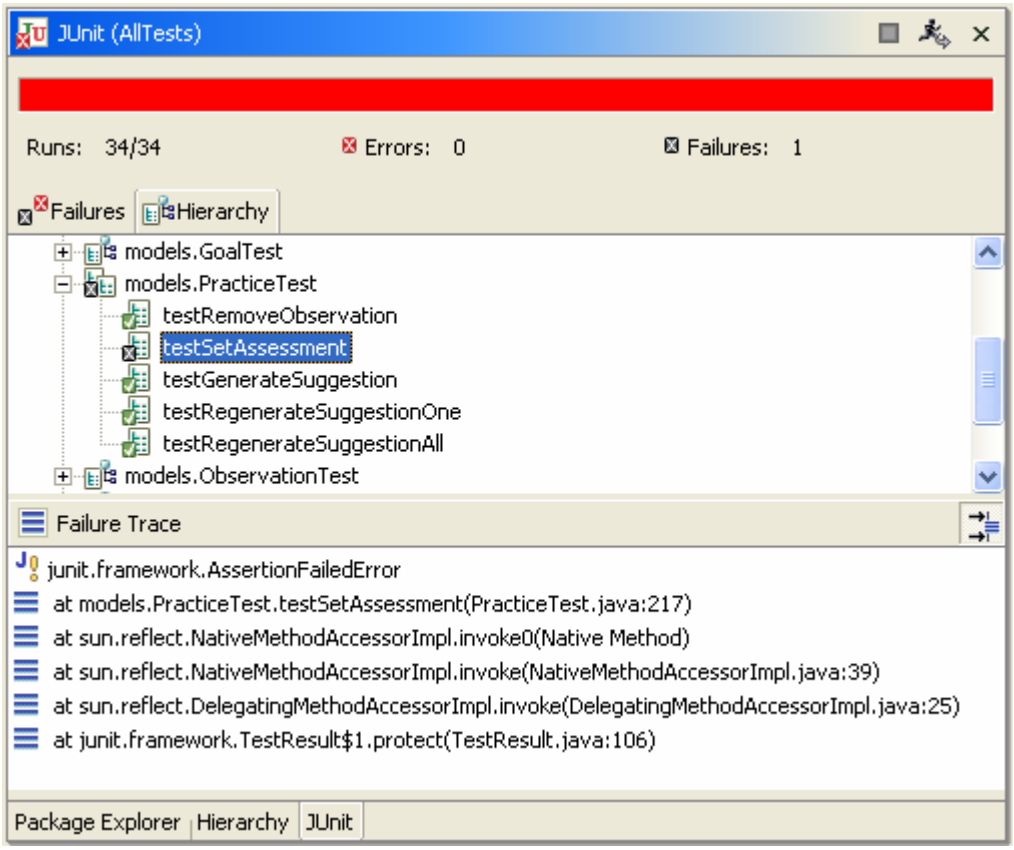

Figura 8.4. Ventana de ejecución de TestCases JUnit.

8.3.2 Pruebas del sistema

Pruebas funcionales

En el siguiente diseño, para cada uno de los casos de uso del sistema se derivan un conjunto de casos de prueba funcionales que ejercitan todos los escenarios del mismo. El diseño comienza contemplando los posibles escenarios a probar dentro del caso de uso, para luego especificar los casos de prueba asociados a cada uno de los escenarios.

De esta manera, el diseño de pruebas funcionales para cada caso de uso se encuentra documentado en dos tablas: una primera tabla que resume los escenarios a

probar, asociándolos a los flujos del caso de uso; y una segunda tabla que lista los casos de prueba para cada uno de los escenarios identificados.

La primera tabla posee el siguiente formato:

Escenario	***Flujos***
Número de escenario	Nombre del flujo o combinación de flujos en el caso de uso

Mientras que el formato de la segunda es el siguiente:

Escenario	***Caso de Prueba***	***Descripción***	***Precondiciones***	***Entradas***	***Salidas***	***Postcondiciones***
Número de escenario	Número de caso de prueba dentro del escenario	Descripción del caso de prueba	Condiciones que deben darse o acciones que deben realizarse antes de la ejecución del caso de prueba	Entradas a suministrar	Salidas a verificar	Condiciones que deben darse o acciones que deben realizarse luego de la ejecución del caso de prueba

A los fines de reducir el esfuerzo demandado por el diseño de las pruebas, y teniendo en cuenta que las pruebas serán realizadas por el tesista, se fusionan los casos de prueba con los procedimientos de prueba, especificando en la misma planilla de casos de prueba todos los detalles necesarios para su ejecución.

Abreviaturas:

Esc: Escenario
CP: Caso de prueba
N2: Nivel 2 del modelo CMMI
AP01: Área de proceso número 1 dentro de un nivel
OE1: Objetivo Específico número 1 dentro de una Área de proceso
P01: Práctica número 1 dentro de un Objetivo

Caso de uso: Administrar evaluaciones

Esc	*Flujos*
1	Flujo básico: Iniciar evaluación
2	Flujo básico: Iniciar evaluación + Alternativa
3	Flujo alternativo: Almacenar evaluación
4	Flujo alternativo: Recuperar evaluación
5	Flujo alternativo: Recuperar evaluación + Alternativa

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba el inicio de una evaluación a nivel de prácticas con instancias	Seleccionar Archivo | Nueva evaluación	**Organización**: Org. de prueba **Alcance**: Prácticas **Instancias**: Proyecto A, Proyecto B (Botón Aceptar)	Se inicializa la ventana principal. A la izquierda debe haber un árbol con raíz “Org. de prueba” y cuatro hojas representando los niveles de madurez (N2 a N5). A la derecha debe haber un panel vacío. En el menú de la aplicación deben habilitarse las siguientes opciones: - Archivo | Guardar evaluación - Edición | Evaluar - Edición | Observaciones	Existe una nueva evaluación en blanco donde se pueden completar las valoraciones Si se selecciona la opción Edición | Evaluar sobre alguno de los niveles, el sistema debe construir el árbol del modelo hasta el nivel de las instancias ingresadas (por ej: N2->AP01->OE1->P01->Proyecto A)
1	2	Prueba el inicio de una evaluación a nivel de prácticas con instancias, dejando las instancias en blanco	Ídem	**Organización**: Org. de prueba **Alcance**: Prácticas **Instancias**: dejar en blanco (Botón Agregar)	El sistema debe mostrar un mensaje indicando que se debe ingresar un nombre de instancia	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	3	Prueba el inicio de una evaluación a nivel de prácticas sin instancias	Ídem	**Organización**: Org. de prueba **Alcance**: Prácticas (Botón Aceptar)	Ídem	Existe una nueva evaluación en blanco donde se pueden completar las valoraciones Si se selecciona la opción Edición | Evaluar sobre alguno de los niveles, el sistema debe construir el árbol del modelo hasta el nivel de las prácticas (por ej: N2->AP01->OE1->P01)
1	4	Prueba el inicio de una evaluación a nivel de objetivos	Ídem	**Organización**: Org. de prueba **Alcance**: Objetivos (Botón Aceptar) (no debe estar disponible la opción Instancias)	Ídem, excepto que no debe habilitarse la opción Edición | Observaciones	Existe una nueva evaluación en blanco donde se pueden completar las valoraciones Si se selecciona la opción Edición | Evaluar sobre alguno de los niveles, el sistema debe construir el árbol del modelo hasta el nivel de los objetivos (por ej: N2->AP01->OE1)
1	5	Prueba el inicio de una evaluación a nivel de áreas de proceso	Ídem	**Organización**: Org. de prueba **Alcance**: Áreas de proceso (Botón Aceptar) (no debe estar disponible la opción Instancias)	Ídem	Existe una nueva evaluación en blanco donde se pueden completar las valoraciones Si se selecciona la opción Edición | Evaluar sobre alguno de los niveles, el sistema debe construir el árbol del modelo hasta el nivel de las áreas de proceso (por ej: N2->AP01)

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	6	Prueba de inicio de evaluación con datos inválidos	Ídem	**Organización**: dejar en blanco (Botón Aceptar)	El sistema debe mostrar un mensaje advirtiendo que debe indicarse una Organización a evaluar.	
1	7	Prueba de inicio de evaluación cancelando la operación	Ídem	(Botón Cancelar)	La ventana principal no se inicializa	No se genera ninguna evaluación.
2	1	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	**1.** Iniciar una evaluación como se especifica en el CP 1.1 **2.** Seleccionar la opción Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Archivo | Nueva evaluación		El sistema debe preguntar si se desea almacenar la evaluación en curso. Seleccionando Sí debe aparecer una ventana para especificar el archivo de destino y luego de guardar la ventana de Inicio de evaluación. Seleccionando No o Si y luego Cancelar, debe aparecer directamente la ventana de Inicio de evaluación.	
2	2	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	Ídem	Repetir los casos de prueba 1.1, 1.2, 1.3 y 1.4	Las especificadas en c/u de los casos de prueba	Las especificadas en c/u de los casos de prueba

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	1	Prueba de almacenamiento de evaluación	**1.** Iniciar una evaluación como se especifica en el CP 1.1 **2.** Seleccionar la opción Edición | Evaluar sobre el nivel N2 **3.** Seleccionar la opción Edición | Evaluar sobre las instancias Proyecto A y Proyecto B de alguna de las prácticas, indicar Valoración elegida en “Completamente Implementada” y Aceptar **4.** Seleccionar la opción Archivo | Guardar evaluación	**Archivo**: c:\temp\prueba.xml (Botón Guardar)		Debe quedar en el disco el archivo c:\temp\prueba.xml Si se selecciona la opción Archivo | Abrir evaluación y se indica ese archivo, debe recuperarse la evaluación en el mismo estado en el que se guardó (con las instancias evaluadas tal como estaban antes de guardar)
3	2	Prueba de almacenamiento de evaluación cancelando la operación	**1.** Iniciar una evaluación como se especifica en el CP 1.1 **2.** Seleccionar la opción Edición | Evaluar sobre el nivel N2 **3.** Seleccionar la opción Archivo | Guardar evaluación	(Botón Cancelar)	La ventana principal permanece inalterada	La evaluación en curso permanece inalterada y se puede seguir adelante con la misma
4	1	Prueba de recuperación de evaluación	**1.** Ejecutar el CP 3.1 **2.** Seleccionar la opción Archivo | Abrir evaluación	**Archivo**: c:\temp\prueba.xml (Botón Abrir)	Se debe recuperar la evaluación en el estado en que estaba al ser guardada, esto es, con las instancias Proyecto A y Proyecto B de la rama N2->AP01->OE1->P01 evaluadas con valoración “Completamente Implementada” y la rama del árbol que las contiene marcada como para revisión (cruz roja)	Se tiene la evaluación recuperada de archivo lista para seguir con las valoraciones.
4	2	Prueba de recuperación de evaluación cancelando la operación	Seleccionar la opción Archivo | Abrir evaluación	(Botón Cancelar)	La ventana principal permanece inalterada	No se recupera ninguna evaluación de archivo
4	3	Prueba de recuperación de evaluación con archivo inexistente	Ídem	**Archivo**: pirulo.txt (Botón Abrir)	El sistema debe mostrar un mensaje advirtiendo que no encuentra el archivo	No se recupera ninguna evaluación de archivo

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
5	1	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	**1.** Iniciar una evaluación como se especifica en el CP 1.1 **2.** Seleccionar la opción Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Archivo | Abrir evaluación		El sistema debe preguntar si se desea almacenar la evaluación en curso. Seleccionando Sí debe aparecer una ventana para especificar el archivo de destino y luego de guardar la ventana de Apertura de evaluación. Seleccionando No o Si y luego Cancelar, debe aparecer directamente la ventana de Apertura de evaluación.	
5	2	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	Ídem	Repetir el caso de prueba 4.1	La especificada en el caso de prueba 4.1	Las especificadas en el caso de prueba 4.1

Caso de uso: Administrar observaciones

Esc	*Flujos*
1	Flujo básico
2	Flujo alternativo: Crear observación
3	Flujo alternativo: Modificar observación
4	Flujo alternativo: Modificar observación + Alternativa
5	Flujo alternativo: Eliminar observación

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba el listado de observaciones existentes cuando no hay observaciones	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Observaciones		Debe abrirse la ventana de administración de observaciones mostrando la lista de observaciones vacía y botones para Agregar, Modificar y Eliminar observaciones.	
2	1	Prueba de creación de observaciones	Ídem	**Tipo**: Debilidad **Impacto**: Significativo **Descripción**: Observación de prueba (Botón Agregar 3 veces)	Se crean 3 observaciones iguales que deben aparecer en la lista de observaciones de la ventana de administración de observaciones. El sistema debe asignarles los números 1, 2 y 3.	Se tienen 3 nuevas observaciones para ser vinculadas a una o más prácticas
2	2	Prueba de creación de observaciones sin datos	Ídem	(Botón Agregar)	Se crea una observación con el Tipo y el Impacto por defecto (Tipo: Debilidad, Impacto: Significativo), y sin ninguna descripción asociada. La observación creada aparece en la lista de observaciones.	Se tiene una nueva observación disponible

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	1	Prueba de modificación de observación	**1.** Iniciar una evaluación con alcance a nivel de prácticas **2.** Crear 3 observaciones con tipo Debilidad e impacto Significativo **3.** Seleccionar Edición | Evaluar sobre el nivel N2 **4.** Seleccionar Edición | Evaluar sobre la práctica AP01->OE1->P01 **5.** Vincular la observación nro 2 a la práctica bajo evaluación **6.** Seleccionar Edición | Observaciones **7.** Seleccionar observación nro 2	**Descripción**: Observación modificada (Botón Modificar)	Se modifica la descripción de la observación nro 2 en la lista de observaciones de la ventana de administración de observaciones y en la lista de observaciones vinculadas de la práctica AP01->OE1->P01	La observación 2 queda modificada
3	2	Prueba de modificación de observación sin alterar datos	Ídem	(Botón Modificar)	No se modifica la observación seleccionada en ninguna de las listas	La observación permanece inalterada

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
4	1	Prueba de modificación de observación con regeneración de valoración en prácticas vinculadas	**1.** Iniciar una evaluación con alcance a nivel de prácticas **2.** Crear 3 observaciones con tipo Debilidad e impacto Significativo **3.** Seleccionar Edición | Evaluar sobre el nivel N2 **4.** Seleccionar Edición | Evaluar sobre la práctica AP01->OE1->P01 **5.** Vincular la observación nro 2 a la práctica bajo evaluación **6.** Definir Artefactos directos: Apropiados, Artefactos Indirectos: Confirman **7.** Asignar Valoración elegida: Ampliamente Implementada **8.** Aceptar **9.** Seleccionar Edición | Observaciones **10.** Seleccionar observación nro 2	**Tipo**: Fortaleza (Botón Modificar)	Se modifica el tipo de la observación en la lista de observaciones de la ventana de administración de observaciones. Se pinta con una cruz roja todo el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Completamente Implementada” Se puede verificar con Edición | Evaluar sobre la práctica AP01->OE1->P01
5	1	Prueba de eliminación de observación	**1.** Iniciar una evaluación con alcance a nivel de prácticas **2.** Crear 3 observaciones con tipo Debilidad e impacto Significativo **3.** Seleccionar Edición | Evaluar sobre el nivel N2 **4.** Seleccionar Edición | Evaluar sobre la práctica AP01->OE1->P01 **5.** Vincular la observación nro 2 a la práctica bajo evaluación **6.** Seleccionar Edición | Observaciones **7.** Seleccionar observación nro 2	(Botón Eliminar)	Se elimina la observación en la lista de observaciones de la ventana de administración de observaciones y en la lista de observaciones vinculadas de la práctica AP01->OE1->P01	La observación desaparece para siempre de la evaluación. Queda un hueco en la numeración de observaciones.

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
5	2	Prueba de eliminación de observación con regeneración de valoración en prácticas vinculadas	**1.** Iniciar una evaluación con alcance a nivel de prácticas **2.** Crear 3 observaciones con tipo Debilidad e impacto Significativo **3.** Seleccionar Edición | Evaluar sobre el nivel N2 **4.** Seleccionar Edición | Evaluar sobre la práctica AP01->OE1->P01 **5.** Vincular la observación nro 2 a la práctica bajo evaluación **6.** Definir Artefactos directos: Apropiados, Artefactos Indirectos: Confirman **7.** Asignar Valoración elegida: Ampliamente Implementada **8.** Aceptar **9.** Seleccionar Edición | Observaciones **10.** Seleccionar observación nro 2	(Botón Eliminar)	Se pinta con una cruz roja todo el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Completamente Implementada” Se puede verificar con Edición | Evaluar sobre la práctica AP01->OE1->P01

Caso de uso: Evaluar niveles de madurez

Esc	*Flujos*
1	Flujo básico
2	Flujo básico + Excepción
3	Flujo alternativo: Evaluación de nivel 3

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba el inicio de la evaluación de nivel de madurez	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2		Debe inicializarse el árbol que representa al modelo en el panel de navegación, mostrando las áreas de proceso, objetivos y prácticas del nivel N2. Debe aparecer la ventana de evaluación del nivel N2 mostrando la guía online y sin ninguna Valoración concluída.	
1	2	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor Alcanzado	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N2	Para todas las Áreas de proceso: **Valoración elegida**: Satisfecho (Botón Aceptar)	La ventana de evaluación del nivel de madurez debe mostrar Valoración concluída: Alcanzado. En el árbol del modelo debe pintarse con una cruz roja el nivel N2.	El nivel N2 cambia su valoración concluída a “Alcanzado” Se puede verificar con Edición | Evaluar sobre el nivel N2

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	3	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor No alcanzado	Ídem	Para una de las Áreas de proceso: **Valoración elegida**: Insatisfecho (Botón Aceptar) Para las restantes: **Valoración elegida**: Satisfecho (Botón Aceptar)	Ídem anterior pero con Valoración concluída: No alcanzado	Ídem anterior pero con Valoración concluída: “No alcanzado”
1	4	Ídem	Ídem	Para una de las Áreas de proceso: **Valoración elegida**: Sin puntaje (Botón Aceptar) Para las restantes: **Valoración elegida**: Satisfecho (Botón Aceptar)	Ídem	Ídem
1	5	Prueba de asignación de valoración al nivel evaluado	Ejecutar el CP 1.2	(Botón Aceptar)	La cruz roja sobre el nivel N2 se convierte en una marca verde	El nivel se encuentra evaluado
2	1	Prueba de evaluación de nivel de madurez 3 sin evaluar niveles inferiores	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N3		El sistema debe mostrar un mensaje de advertencia diciendo que antes de evaluar un nivel deben haberse alcanzado todos los niveles inferiores. Al aceptar la advertencia se abre la ventana de evaluación de nivel de madurez y el sistema permite efectuar la evaluación.	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
2	2	Prueba de evaluación de nivel de madurez 4 sin evaluar niveles inferiores	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N4		Ídem	
2	3	Prueba de evaluación de nivel de madurez 5 sin evaluar niveles inferiores	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N5		Ídem	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
2	4	Prueba de evaluación de nivel de madurez con evaluación de nivel inferior	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N2, y asignar Valoración sugerida: Satisfecho a todas las áreas. **4.** Seleccionar Edición | Evaluar sobre el nivel N3 **5.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N3, y asignar Valoración sugerida: Satisfecho a todas las áreas. **6.** Seleccionar Edición | Evaluar sobre el nivel N4 **7.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N4, y asignar Valoración sugerida: Satisfecho a todas las áreas. **8.** Seleccionar Edición | Evaluar sobre el nivel N5		Debe inicializarse el árbol que representa al modelo en el panel de navegación, mostrando las áreas de proceso del nivel N5. Debe aparecer la ventana de evaluación del nivel N5 mostrando la guía online y sin ninguna Valoración concluída.	Los niveles 2 a 4 se encuentran alcanzados y se puede proceder a la evaluación del nivel 5

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	1	Prueba de evaluación de nivel 3 con alcance prácticas y nivel 2 alcanzado	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso, objetivos y prácticas del nivel N2, asignando Valoración sugerida: Satisfecho o Completamente Implementado según corresponda. **4.** Seleccionar Edición | Evaluar sobre el nivel N3		El árbol de navegación del modelo debe marcar con una cruz roja al nivel 2. Abriendo las áreas de proceso, se debe verificar que los objetivos genéricos se reemplazaron por los de nivel 3 y superiores. Las valoraciones existentes para los objetivos genéricos y sus prácticas se pierden.	El nivel 2 pierde todas las evaluaciones de sus objetivos genéricos. El nivel 3 se encuentra en condiciones de ser evaluado.
3	2	Prueba de evaluación de nivel 3 con alcance objetivos y nivel 2 alcanzado	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso y objetivos del nivel N2, asignando Valoración sugerida: Satisfecho. **4.** Seleccionar Edición | Evaluar sobre el nivel N3		Debe inicializarse el árbol que representa al modelo en el panel de navegación, mostrando las áreas de proceso del nivel N3. Debe aparecer la ventana de evaluación del nivel N3 mostrando la guía online y sin ninguna Valoración concluída.	Se puede proceder a la evaluación del nivel 3

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	3	Prueba de evaluación de nivel 3 con alcance áreas de proceso y nivel 2 alcanzado	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N2, asignando Valoración sugerida: Satisfecho. **4.** Seleccionar Edición | Evaluar sobre el nivel N3		Debe inicializarse el árbol que representa al modelo en el panel de navegación, mostrando las áreas de proceso del nivel N3. Debe aparecer la ventana de evaluación del nivel N3 mostrando la guía online y sin ninguna Valoración concluída.	Se puede proceder a la evaluación del nivel 3

Caso de uso: Evaluar áreas de proceso

Esc	*Flujos*
1	Flujo básico
2	Flujo básico + Excepción
3	Flujo alternativo: Alcance hasta nivel de áreas de proceso

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Satisfecho	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada uno de los objetivos del área de proceso AP01	Para todos los Objetivos: **Valoración elegida**: Satisfecho (Botón Aceptar) Luego de evaluar todos los objetivos, seleccionar Edición | Evaluar sobre el área de proceso AP01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Satisfecho. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01	El área de proceso N2->AP01 cambia su valoración sugerida a “Satisfecho”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	2	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Insatisfecho	Ídem	Para uno de los Objetivos: **Valoración elegida**: Insatisfecho (Botón Aceptar) Para los restantes: **Valoración elegida**: Satisfecho (Botón Aceptar) Luego de evaluar todos los objetivos, seleccionar Edición | Evaluar sobre el área de proceso AP01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Insatisfecho. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01	El área de proceso N2->AP01 cambia su valoración sugerida a “Insatisfecho”
1	3	Prueba de asignación de valoración elegida al área de proceso evaluada	Ejecutar el CP 1.1	**Valoración elegida**: Satisfecho (Botón Aceptar)	La cruz roja sobre el área de proceso N2->AP01 se convierte en una marca verde	El área de proceso se encuentra evaluada
1	4	Prueba de asignación de valoración elegida sin asignar valor	Ídem	(Botón Aceptar)	El sistema debe mostrar un mensaje indicando que se debe indicar una Valoración elegida	
2	1	Prueba de evaluación de área de proceso sin evaluar objetivos	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar el área de proceso AP01		El sistema debe mostrar un mensaje de advertencia diciendo que antes de evaluar un área de proceso deben evaluarse todos sus objetivos. Al aceptar la advertencia se abre la ventana de evaluación de áreas de proceso pero sin la posibilidad de asignar valoraciones.	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
2	2	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Fuera de alcance	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar el área de proceso AP01 **4.** Aceptar el mensaje de advertencia	**Fuera de alcance**: Sí	La ventana de evaluación de áreas de proceso debe mostrar Valoración sugerida: No aplicable Deben habilitarse la lista de Valoraciones elegidas, la Justificación y el botón Aceptar	Puede asignarse una valoración elegida al área de proceso sin evaluar los objetivos de la misma.
2	3	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Sin puntaje	Ídem	**Sin evidencia objetiva**: Sí	La ventana de evaluación de áreas de proceso debe mostrar Valoración sugerida: Sin puntaje Deben habilitarse la lista de Valoraciones elegidas, la Justificación y el botón Aceptar	Puede asignarse una valoración elegida al área de proceso sin evaluar los objetivos de la misma.
3	1	Prueba de evaluación con alcance áreas de proceso	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar el área de proceso AP01		Se abre la ventana de evaluación de áreas de proceso sin ninguna Valoración sugerida	
3	2	Prueba de asignación de valoración elegida con alcance a nivel de áreas de proceso	Ídem	**Valoración elegida**: Satisfecho (Botón Aceptar)	Se pinta con una marca verde el área de proceso N2->AP01 en el árbol de navegación del modelo	El área de proceso se encuentra evaluada

Caso de uso: Evaluar objetivos

Esc	*Flujos*
1	Flujo básico
2	Flujo básico + Excepción
3	Flujo alternativo: Alcance hasta nivel de objetivos

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Satisfecho	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las prácticas del objetivo N2->AP01->OE1	Para 4 Prácticas: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Para las Prácticas restantes: **Valoración elegida**: Ampliamente Implementada (Botón Aceptar) Luego de evaluar todas las prácticas, seleccionar Edición | Evaluar sobre el objetivo N2->AP01->OE1	Debe abrirse la ventana de evaluación de objetivos mostrando Valoración sugerida: Satisfecho. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1	El objetivo N2->AP01->OE1 cambia su valoración sugerida a “Satisfecho”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	2	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho	Ídem	Para una de las Prácticas: **Valoración elegida**: Parcialmente Implementada (Botón Aceptar) Para otra de las Prácticas: **Valoración elegida**: No Implementada (Botón Aceptar) Para las restantes: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Luego de evaluar todas las prácticas, seleccionar Edición | Evaluar sobre el objetivo N2->AP01->OE1	Debe abrirse la ventana de evaluación de objetivos mostrando Valoración sugerida: Insatisfecho. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1	El objetivo N2->AP01->OE1 cambia su valoración sugerida a “Insatisfecho”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	3	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho por observaciones tipo Debilidad	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Crear 3 observaciones con tipo Debilidad e impacto Significativo **4.** Seleccionar Edición | Evaluar sobre cada una de las prácticas del objetivo N2->AP01->OE1	Para una de las Prácticas: **Valoración elegida**: Ampliamente Implementada **Observaciones**: las 3 observaciones tipo Debilidad (Botón Aceptar) Para las restantes: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Luego de evaluar todas las prácticas, seleccionar Edición | Evaluar sobre el objetivo N2->AP01->OE1	Debe abrirse la ventana de evaluación de objetivos mostrando Valoración sugerida: Insatisfecho. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1	El objetivo N2->AP01->OE1 cambia su valoración sugerida a “Insatisfecho”
1	4	Prueba de asignación de valoración elegida al objetivo evaluado	Ejecutar el CP 1.1	**Valoración elegida**: Satisfecho (Botón Aceptar)	La cruz roja sobre el objetivo N2->AP01->OE1 se convierte en una marca verde	El objetivo se encuentra evaluado
1	5	Prueba de asignación de valoración elegida sin asignar valor	Ídem	(Botón Aceptar)	El sistema debe mostrar un mensaje indicando que se debe indicar una Valoración elegida	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
2	1	Prueba de evaluación de objetivo sin evaluar prácticas	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar el objetivo N2->AP01->OE1		El sistema debe mostrar un mensaje de advertencia diciendo que antes de evaluar un objetivo deben evaluarse todas sus prácticas Al aceptar la advertencia se abre la ventana de evaluación de objetivos pero sin la posibilidad de asignar valoraciones.	
3	1	Prueba de evaluación con alcance objetivos	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar el objetivo N2->AP01->OE1		Se abre la ventana de evaluación de objetivos sin ninguna Valoración sugerida	
3	2	Prueba de asignación de valoración elegida con alcance a nivel de objetivos	Ídem	**Valoración elegida**: Satisfecho (Botón Aceptar)	Se pinta con una marca verde el objetivo N2->AP01->OE1 en el árbol de navegación del modelo	El objetivo se encuentra evaluado

Caso de uso: Evaluar Prácticas

Esc	*Flujos*
1	Flujo básico
2	Flujo básico + Excepción
3	Flujo alternativo: No se definieron múltiples instancias

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Completamente Implementada	**1.** Iniciar una evaluación con alcance a nivel de Prácticas y 2 instancias (Proyecto A y Proyecto B) **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las instancias de la práctica N2->AP01->OE1->P01	Para todas las instancias: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Completamente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Completamente Implementada”
1	2	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Ídem	Para todas las instancias: **Valoración elegida**: Ampliamente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Ampliamente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Ampliamente Implementada”
1	3	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Parcialmente Implementada	Ídem	Para todas las instancias: **Valoración elegida**: Parcialmente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Parcialmente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Parcialmente Implementada”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	4	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada	Ídem	Para todas las instancias: **Valoración elegida**: No Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: No Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “No Implementada”
1	5	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Ídem	Para una instancia: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Para la otra instancia: **Valoración elegida**: Ampliamente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Ampliamente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Ampliamente Implementada”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	6	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada o Parcialmente Implementada	Ídem	Para una instancia: **Valoración elegida**: Parcialmente Implementada (Botón Aceptar) Para la otra instancia: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: Ampliamente o Parcialmente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “Ampliamente o Parcialmente Implementada”
1	7	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada, Parcialmente Implementada o Ampliamente Implementada	Ídem	Para una instancia: **Valoración elegida**: No Implementada (Botón Aceptar) Para la otra instancia: **Valoración elegida**: Completamente Implementada (Botón Aceptar) Luego de evaluar todas las instancias, seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	Debe abrirse la ventana de evaluación de áreas de proceso mostrando Valoración sugerida: No Implementada, Parcialmente Implementada o Ampliamente Implementada. En el árbol de navegación del modelo se pinta con una cruz roja el camino N2->AP01->OE1->P01	La práctica N2->AP01->OE1->P01 cambia su valoración sugerida a “No Implementada, Parcialmente Implementada o Ampliamente Implementada”

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	8	Prueba de asignación de valoración elegida a la práctica evaluada	Ejecutar el CP 1.1	**Valoración elegida**: Completamente Implementada (Botón Aceptar)	La cruz roja sobre la práctica N2->AP01->OE1->P01 se convierte en una marca verde	La práctica se encuentra evaluada
1	9	Prueba de asignación de valoración elegida sin asignar valor	Ídem	(Botón Aceptar)	El sistema debe mostrar un mensaje indicando que se debe indicar una Valoración elegida	
2	1	Prueba de evaluación de práctica sin evaluar instancias	**1.** Iniciar una evaluación con alcance a nivel de Prácticas y 2 instancias (Proyecto A y Proyecto B) **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar la práctica N2->AP01->OE1->P01		El sistema debe mostrar un mensaje de advertencia diciendo que antes de evaluar una práctica a nivel conjunto deben evaluarse todos las instancias	
3	1	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Completamente Implementada	**1.** Iniciar una evaluación con alcance a nivel de Prácticas sin instancias **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar la práctica N2->AP01->OE1->P01	**Artefactos directos**: Apropiados **Artefactos Indirectos**: Confirman **Observaciones**: ninguna	La ventana de evaluación de prácticas muestra Valoración sugerida: Completamente Implementada	
3	2	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Ampliamente Implementada	**1.** Iniciar una evaluación con alcance a nivel de Prácticas sin instancias **2.** Crear 3 observaciones tipo Debilidad con impacto Significativo **3.** Seleccionar Edición | Evaluar sobre el nivel N2 **4.** Seleccionar Edición | Evaluar la práctica N2->AP01->OE1->P01	**Artefactos directos**: Apropiados **Artefactos Indirectos**: Confirman **Observaciones**: vincular las 3 observaciones (Botón Aceptar)	La ventana de evaluación de prácticas muestra Valoración sugerida: Ampliamente Implementada	

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
3	3	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Parcialmente Implementada	Ídem	**Artefactos directos**: No Apropiados **Artefactos Indirectos**: Sugieren **Observaciones**: vincular las 3 observaciones (Botón Aceptar)	La ventana de evaluación de prácticas muestra Valoración sugerida: Parcialmente Implementada	
3	4	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor No Implementada	Ídem	**Artefactos directos**: No se sabe **Artefactos Indirectos**: No se sabe **Observaciones**: ninguna (Botón Aceptar)	La ventana de evaluación de prácticas muestra Valoración sugerida: No Implementada	
3	5	Prueba de asignación de valoración elegida con alcance a nivel prácticas sin instancias	Ídem	**Valoración elegida**: Completamente Implementada (Botón Aceptar)	Se pinta con una marca verde la práctica N2->AP01->OE1->P01 en el árbol de navegación del modelo	La práctica se encuentra evaluada

Caso de uso: Generar reporte

Esc	*Flujos*
1	Flujo básico

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	1	Prueba la generación de un reporte para alcance a nivel de Áreas de proceso	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las áreas de proceso del nivel N2 **4.** Indicar **Valoración elegida**: Satisfecho y **Justificación**: Ninguna en todas las áreas **5.** Aceptar la valoración concluida para el nivel N2. **6.** Seleccionar Archivo | Generar reporte		Se abre la ventana de previsualización del reporte mostrando el reporte generado. El reporte muestra un encabezado con los datos de la **Organización** y el **Alcance de la evaluación** en todas las páginas. Para el Nivel de madurez N2 muestra **Valoración sugerida**: Alcanzado y **Valoración elegida**: Alcanzado. Debajo e indentado a la derecha muestra las áreas de proceso AP01 a AP07 con **Valoración sugerida** en blanco, **Valoración elegida**: Satisfecho, y **Justificación**: Ninguna	El reporte generado está disponible para que el usuario lo imprima. Seleccionando el botón de la parte superior izquierda se puede imprimir el reporte.

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	2	Prueba de campo Justificación grande (múltiples renglones)	**1.** Iniciar una evaluación con alcance a nivel de Áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre el área de proceso AP01 **4.** Indicar **Valoración elegida**: Satisfecho y **Justificación**: aaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbb ccccccccccccccccccccccccc **6.** Seleccionar Archivo | Generar reporte		La ventana de previsualización muestra los siguientes datos para el área de proceso AP01: **Valoración sugerida**: (en blanco) **Valoración elegida**: Satisfecho **Justificación**: aaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbb ccccccccccccccccccccccccc	Ídem
1	3	Prueba de campo Justificación grande (que exceda la longitud de un renglón)	Ídem, pero llenar el campo **Justificación** presionando la tecla 'a' durante unos segundos (de manera que se acumulen un par de cientos de repeticiones)		La ventana de previsualización muestra los siguientes datos para el área de proceso AP01: **Valoración sugerida**: (en blanco) **Valoración elegida**: Satisfecho **Justificación**: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaa (tantas veces como repeticiones se ingresaron)	Ídem

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	4	Prueba la generación de un reporte para alcance a nivel de Objetivos	**1.** Iniciar una evaluación con alcance a nivel de Objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada uno de los objetivos del área de proceso AP01 **4.** Indicar **Valoración elegida**: Satisfecho y **Justificación**: Ninguna en todos los objetivos **5.** Seleccionar Archivo | Generar reporte		Se abre la ventana de previsualización del reporte mostrando el reporte generado. El reporte muestra un encabezado con los datos de la **Organización** y el **Alcance de la evaluación** en todas las páginas. Para el Nivel de madurez N2 muestra **Valoración sugerida** y **Valoración elegida** ambas en blanco Para el Área de proceso AP01 muestra **Valoración sugerida**: Satisfecho, **Valoración elegida** y **Justificación** en blanco. Para los objetivos OE1 y OG del Área de proceso AP01 muestra **Valoración sugerida** en blanco, **Valoración elegida**: Satisfecho, y **Justificación**: Ninguna	El reporte generado está disponible para que el usuario lo imprima. Seleccionando el botón de la parte superior izquierda se puede imprimir el reporte.

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	5	Prueba la generación de un reporte para alcance a nivel de Prácticas	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre cada una de las prácticas del objetivo OE1 del área de proceso AP01 **4.** Indicar **Valoración elegida**: Completamente Implementado y **Justificación**: Ninguna en todas las prácticas **5.** Seleccionar Archivo | Generar reporte		Se abre la ventana de previsualización del reporte mostrando el reporte generado. El reporte muestra un encabezado con los datos de la **Organización** y el **Alcance de la evaluación** en todas las páginas. Para el Área de proceso AP01 muestra **Valoración sugerida**, **Valoración elegida** y **Justificación** en blanco. Para el Área de proceso AP01 muestra **Valoración sugerida**, **Valoración elegida** y **Justificación** en blanco. Para el objetivo OE1 del Área de proceso AP01 muestra **Valoración sugerida**: Satisfecho, **Valoración elegida** y **Justificación** en blanco Para las áreas P01 a P05 del objetivo OE1, muestra **Valoración sugerida** en blando, **Valoración elegida**: Completamente Implementado, y **Justificación**: Ninguna	El reporte generado está disponible para que el usuario lo imprima. Seleccionando el botón de la parte superior izquierda se puede imprimir el reporte.

Esc	*CP*	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*	*Postcondiciones*
1	6	Prueba de generación de reporte a nivel de Prácticas con todos los niveles evaluados	**1.** Iniciar una evaluación con alcance a nivel de Prácticas **2.** Seleccionar Edición | Evaluar sobre los niveles N2 a N5 **6.** Seleccionar Archivo | Generar reporte		Se abre la ventana de previsualización del reporte mostrando el reporte generado. El reporte muestra un encabezado con los datos de la **Organización** y el **Alcance de la evaluación** en todas las páginas. Para los niveles de madurez N2 a N5 muestra **Valoración sugerida** y **Valoración elegida** ambas en blanco Para todas las Áreas de proceso, Objetivos y Prácticas, muestra **Valoración sugerida**, **Valoración elegida** y **Justificación** en blanco.	Ídem

Pruebas de rendimiento

Aunque en las especificaciones del sistema no se han detallado requerimientos específicos asociados al rendimiento, se considera que los tiempos de respuesta reducidos y el buen desempeño del sistema son vitales para el funcionamiento aceptable del mismo.

Por tal motivo, en el siguiente diseño se derivan un conjunto de casos de prueba que ejercitan algunos escenarios donde es importante el rendimiento del sistema.

Los casos de prueba se documentan en una tabla con el siguiente formato:

Caso de Prueba	***Descripción***	***Precondiciones***	***Entradas***	***Salidas***
Número de caso de prueba	Descripción del caso de prueba	Condiciones que deben darse o acciones que deben realizarse antes de la ejecución del caso de prueba	Entradas a suministrar	Salidas a verificar

A los fines de reducir el esfuerzo demandado por el diseño de las pruebas, y teniendo en cuenta que las pruebas serán realizadas por el tesista, se fusionan los casos de prueba con los procedimientos de prueba, especificando en la misma planilla de casos de prueba todos los detalles necesarios para su ejecución.

Abreviaturas:

CP: Caso de prueba
N2: Nivel 2 del modelo CMMI
AP01: Área de proceso número 1 dentro de un nivel
OE1: Objetivo Específico número 1 dentro de una Área de proceso
P01: Práctica número 1 dentro de un Objetivo

CP	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*
1	Prueba el inicio de una evaluación a nivel de Prácticas con instancias	Seleccionar Archivo | Nueva evaluación	**Organización**: Org. de prueba **Alcance**: Prácticas **Instancias**: Proyecto A, Proyecto B (Botón Aceptar)	Se debe inicializar el escritorio de la aplicación (árbol con el modelo a la izquierda y área para las ventanas de evaluación a la derecha) en un tiempo menor o igual a 1 segundo.
2	Prueba de evaluación de Nivel de madurez	**1.** Ejecutar el CP 1 **2.** Seleccionar Edición | Evaluar sobre el nivel N2		Se debe inicializar toda la rama que cuelga del nivel N2, y debe aparecer la ventana de Evaluación de nivel de madurez, en un tiempo menor o igual a 2 segundos.
3	Prueba de inferencia de valoraciones a nivel de instancia	**1.** Ejecutar el CP 2 **2.** Seleccionar Edición | Evaluar sobre la instancia N2->AP01->OE1->P01->Proyecto A	**Artefactos directos**: Apropiados **Artefactos indirectos**: Confirman	La ventana debe mostrar una Valoración sugerida en un tiempo menor a 1 segundo.
4	Prueba de asignación de valoración elegida a nivel de instancia	Ídem	**Valoración elegida**: Completamente Implementada	Se debe pintar con una marca verde la instancia N2->AP01->OE1->P01->Proyecto A, y debe cerrarse la ventana de evaluación. Todo debe ocurrir en un tiempo menor o igual a 1 segundo.
5	Prueba de inferencia de valoraciones a nivel conjunto de instancias	**1.** Ejecutar el CP 2 **2.** Seleccionar Edición | Evaluar sobre la instancia N2->AP01->OE1->P01->Proyecto A **3.** Asignar Valoración elegida: Completamente Implementada **4.** Seleccionar Edición | Evaluar sobre la instancia N2->AP01->OE1->P01->Proyecto B **5.** Asignar Valoración elegida: Completamente Implementada		Se debe pintar con una cruz roja el camino N2->AP01->OE1->P01, indicando que ha cambiado la valoración sugerida para P01. Todo debe ocurrir en un tiempo menor o igual a 1 segundo

CP	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*
6	Prueba de inferencia de valoraciones a nivel de Práctica sin instancias	**1.** Iniciar una evaluación a nivel de prácticas sin instancias **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01	**Artefactos directos**: Apropiados **Artefactos indirectos**: Confirman	La ventana debe mostrar una Valoración sugerida en un tiempo menor a 1 segundo.
7	Prueba de asignación de valoración elegida a nivel de Práctica sin instancias	Ídem	**Valoración elegida**: Completamente Implementada	Se debe pintar con una marca verde la práctica N2->AP01->OE1->P01, y debe cerrarse la ventana de evaluación. Todo debe ocurrir en un tiempo menor o igual a 1 segundo.
8	Prueba de inferencia de valoraciones a nivel Objetivo	**1.** Iniciar una evaluación a nivel de prácticas sin instancias **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01 y asignar Valoración elegida: Completamente Implementada **4.** Repetir el paso anterior para las prácticas P02, P03, P04 y P05 del objetivo OE1		Se debe pintar con una cruz roja el camino N2->AP01->OE1, indicando que ha cambiado la valoración sugerida para OE1. Todo debe ocurrir en un tiempo menor o igual a 1 segundo
9	Prueba de inferencia de valoraciones a nivel Área de proceso	**1.** Iniciar una evaluación a nivel de objetivos **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre el objetivo N2->AP01->OE1 y asignar Valoración elegida: Satisfecho **4.** Repetir el paso anterior para el objetivo OG del área de proceso AP01		Se debe pintar con una cruz roja el camino N2->AP01, indicando que ha cambiado la valoración sugerida para AP01. Todo debe ocurrir en un tiempo menor o igual a 1 segundo
10	Prueba de inferencia de valoraciones a nivel Nivel de madurez	**1.** Iniciar una evaluación a nivel de áreas de proceso **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Seleccionar Edición | Evaluar sobre el área de proceso N2->AP01 y asignar Valoración elegida: Satisfecho **4.** Repetir el paso anterior para las áreas de proceso AP02, AP03, AP04, AP05, AP06 y AP07 del nivel de madurez N2		Se debe pintar con una cruz roja el camino N2, indicando que ha cambiado la valoración sugerida para N2. Todo debe ocurrir en un tiempo menor o igual a 1 segundo

CP	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*
11	Prueba de almacenamiento de evaluación en curso	**1.** Iniciar una evaluación a nivel de prácticas con 2 instancias (Proyecto A y Proyecto B) **2.** Seleccionar Edición | Evaluar sobre los niveles N2 a N5 **3.** Seleccionar Edición | Evaluar sobre la instancia N2->AP01->OE1->P01->Proyecto A y asignar Valoración elegida: Completamente Implementado **4.** Repetir para Proyecto B **5.** Repetir los pasos 3 y 4 para los niveles N3, N4 y N5 **2.** Seleccionar Archivo | Guardar Evaluación	**Archivo**: c:\temp\prueba.xml	El sistema debe guardar la evaluación en el archivo indicado y cerrar la ventana de almacenamiento de evaluación. Todo debe ocurrir en un tiempo menor o igual a 5 segundos.
12	Prueba de apertura de evaluación almacenada	**1.** Ejecutar el CP 11 **2.** Seleccionar Archivo | Abrir Evaluación	**Archivo**: c:\temp\prueba.xml	El sistema debe recuperar la evaluación almacenada, cerrar la ventana de apertura de evaluación, e inicializar el escritorio de la aplicación. Todo debe ocurrir en un tiempo menor o igual a 5 segundos.
13	Prueba de propagación de modificaciones en observaciones	**1.** Iniciar una evaluación a nivel de prácticas **2.** Seleccionar Edición | Evaluar sobre el nivel N2 **3.** Crear 3 observaciones tipo Debilidad impacto Significativo **4.** Seleccionar Edición | Evaluar sobre la práctica N2->AP01->OE1->P01 **5.** Vincular la observación nro 2 a la práctica, e indicar Artefactos Directos: Apropiados y Artefactos Indirectos: Confirman (lo que genera como Valoración sugerida: Ampliamente Implementado) **6.** Seleccionar Edición | Observaciones **7.** Modificar observación nro 2, cambiando el tipo a Fortaleza		El sistema debe refrescar la observación modificada en la lista de observaciones vinculadas de la ventana de evaluación de prácticas, y debe generar una nueva valoración sugerida con valor Completamente Implementado. Todo debe ocurrir en un tiempo menor o igual a 1 segundo.

CP	*Descripción*	*Precondiciones*	*Entradas*	*Salidas*
14	Prueba de generación de reporte	**1.** Iniciar una evaluación a nivel de prácticas **2.** Seleccionar Edición | Evaluar sobre los niveles N2 a N5 **3.** Seleccionar Archivo | Generar reporte		El sistema debe mostrar la ventana de previsualización de reportes con el reporte generado, en un tiempo menor o igual a 5 segundos.

Pruebas de facilidad de uso

No se defininen casos de prueba formales para este tipo de pruebas. Queda a cargo del usuario la utilización del sistema y la evaluación de la usabilidad del mismo.

8.3.3 Pruebas de implantación y aceptación

Los casos de prueba a utilizar en este tipo de pruebas son los mismos que se encuentran documentados en el apartado **8.3.2 (Pruebas del sistema)**.

9. CONSTRUCCIÓN DEL SISTEMA

En este capítulo se documentan los productos de salida del proceso CSI (Construcción del Sistema de Información) de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

9.1 Entorno de construcción

Como primer paso para la construcción del sistema, se preparó el entorno de acuerdo a lo especificado en el apartado 7.8.1 (Especificación del entorno de construcción) del capítulo 7.

La preparación del entorno consistió de los siguientes pasos:

1. Instalación de la plataforma J2SE versión 1.4.2, la cual incluye el entorno integrado de desarrollo Netbeans 3.5 [J2SE, 2004]
2. Instalación y configuración del entorno integrado de desarrollo Eclipse 2.1.1 [Eclipse, 2004]
3. Incorporación de la librería JUnit 3.8.1 al entorno Eclipse [JUnit, 2004]
4. Instalación y configuración de la herramienta CASE Enterprise Architect 3.6 [EA, 2004]

9.2 Código fuente de los componentes

Una vez preparado el entorno de construcción, se procedió a generar el código fuente de cada uno de los componentes contemplados en el diseño de la aplicación, siguiendo el orden establecido en el apartado 7.8.3 (Plan de integración del sistema) del capítulo 7.

Para cada clase del modelo de diseño, se generó el código fuente en lenguaje Java desde la herramienta *Enterprise Architect*. Posteriormente, el código de las clases se completó utilizando el entorno *Eclipse*, agregando el comportamiento especificado en los diagramas de secuencia.

Los componentes del paquete "*views*" (interfaces gráficas de usuario) se generaron íntegramente en el entorno *Netbeans*, el cual brinda la posibilidad de construir las interfaces de manera gráfica, generando automáticamente el código fuente en lenguaje Java.

Finalmente, las clases se integraron entre sí y se efectuaron las pruebas unitarias y de integración.

9.3 Pruebas unitarias y de integración

Las pruebas unitarias y de integración se efectuaron en paralelo con la generación del código de las clases que conforman el sistema. Esto significa que a medida que se escribía el código, se ejecutaban las pruebas a manera de regresión, garantizando que el nuevo código no introducía errores sobre el código existente.

Para cada uno de los componentes que conforman el sistema se ejecutaron los casos de prueba unitarios y de integración especificados en el apartado 8.3.1 (Pruebas unitarias y de integración) del capítulo 8.

Todos los defectos encontrados fueron corregidos durante el desarrollo.

La tabla 9.1 muestra los resultados de la ejecución de dichas pruebas luego de finalizadas las correcciones.

CP	***Componente***	***Descripción***	***Resultado***
1	Subsistema *persistence*	Prueba método *write* del *PersistenceManager*. Genera un objeto X de prueba y lo almacena mediante el *PersistenceManager*.	Correcto
2	Subsistema *persistence*	Prueba método *read* del *PersistenceManager*. Genera un objeto X de prueba y lo compara con otro leído de archivo por medio del *PersistenceManager*	Correcto
3	Subsistema *messages*	Prueba método *getMessage* del *MessageManager*. Inicializa el *MessageManager* para un idioma determinado (español) y recupera un mensaje de prueba con un argumento asociado	Correcto
4	Clase *Appraisal*	Prueba método *setSaved*.	Correcto

CP	Componente	Descripción	Resultado
5	Clase *Appraisal*	Prueba método *addObservation*.	Correcto
6	Clase *Appraisal*	Prueba método *updateObservation*.	Correcto
7	Clase *Appraisal*	Prueba método *removeObservation*.	Correcto
8	Clase *Appraisal*	Prueba método *setDirty*.	Correcto
9	Clase *Appraisal*	Prueba método *addChild*. Setea un nivel de madurez como hijo de una evaluación. Luego pregunta a la evaluación por el hijo recién agregado.	Correcto
10.1	Clase *MaturityLevel*	Prueba método *regenerateSuggestion*. Setea un par de áreas de proceso cualquiera como hijas de un nivel de madurez cualquiera y varía la valoración elegida de cada una de ellas, de manera de abarcar todas las reglas de inferencia.	Correcto
10.2	Ídem anterior	Ídem anterior	Correcto
10.3	Ídem anterior	Ídem anterior	Correcto
10.4	Ídem anterior	Ídem anterior	Correcto
11	Clase *MaturityLevel*	Prueba método *setDirty*. Setea un nivel de madurez como hijo de una evaluación, e invoca a setDirty para vericar que la notificación se propague al padre.	Correcto
12	Clase *MaturityLevel*	Prueba método *addChild*. Setea un área de proceso como hija de un nivel de madurez. Luego pregunta al nivel de madurez por el hijo recién agregado.	Correcto
13.1	Clase *ProcessArea*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor del área de proceso cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	Correcto
13.2	Ídem anterior	Ídem anterior	Correcto
13.3	Ídem anterior	Ídem anterior	Correcto
13.4	Ídem anterior	Ídem anterior	Correcto
14.1	Clase *ProcessArea*	Prueba método *regenerateSuggestion*. Setea un par de objetivos como hijos del área de proceso y varía la valoración elegida de cada una de ellos, de manera de barrer todas las reglas de inferencia.	Correcto
14.2	Ídem anterior	Ídem anterior	Correcto
14.3	Ídem anterior	Ídem anterior	Correcto
14.4	Ídem anterior	Ídem anterior	Correcto
14.5	Ídem anterior	Ídem anterior	Correcto
15.1	Clase *ProcessArea*	Prueba método *generateSuggestion*. Invoca al método con distintos valores de Fuera de alcance y Sin evidencia, de manera de ejercitar todas las reglas de inferencia.	Correcto
15.2	Ídem anterior	Ídem anterior	Correcto
16.1	Clase *Goal*	Prueba método *regenerateSuggestion*. Setea un par de prácticas como hijas del objetivo y varía la valoración elegida y las observaciones de cada una de ellas, de manera de barrer todas las reglas de inferencia.	Correcto
16.2	Ídem anterior	Ídem anterior	Correcto
16.3	Ídem anterior	Ídem anterior	Correcto
16.4	Ídem anterior	Ídem anterior	Correcto

CP	Componente	Descripción	Resultado
17.1	Clase *Goal*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor del objetivo cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	Correcto
17.2	Ídem anterior	Ídem anterior	Correcto
18.1	Clase *Practice*	Prueba método *regenerateSuggestion* para el caso de práctica a nivel de instancia. Setea valores para Artefactos directos, Artefactos indirectos y observaciones de manera de ejercitar todas las reglas de inferencia.	Correcto
18.2	Ídem anterior	Ídem anterior	Correcto
18.3	Ídem anterior	Ídem anterior	Correcto
18.4	Ídem anterior	Ídem anterior	Correcto
19.1	Clase *Practice*	Prueba método *regenerateSuggestion* para el caso de práctica a nivel de conjunto de instancias. Asigna dos instancias como hijas de una práctica, y setea diferentes valores de valoración a cada una de las instancias para ejercitar todas las reglas de inferencia.	Correcto
19.2	Ídem anterior	Ídem anterior	Correcto
19.3	Ídem anterior	Ídem anterior	Correcto
19.4	Ídem anterior	Ídem anterior	Correcto
19.5	Ídem anterior	Ídem anterior	Correcto
20	Clase *Practice*	Prueba método *removeObservation*. Setea valores para Artefactos directos, Artefactos indirectos y observaciones de manera de obtener una valoración sugerida. A continuación elimina una observación provocando un cambio en la valoración sugerida.	Correcto
21.1	Clase *Practice*	Prueba método *generateSuggestion*. Invoca al método con distintos valores de Artefactos directos, Artefactos indirectos y Observaciones de manera de ejercitar todas las reglas de inferencia.	Correcto
21.2	Ídem anterior	Ídem anterior	Correcto
21.3	Ídem anterior	Ídem anterior	Correcto
21.4	Ídem anterior	Ídem anterior	Correcto
22.1	Clase *Practice*	Prueba método *setAssessment*. Setea distintos valores en la valoración elegida, verificando que se notifique al contenedor de la práctica cuando hay un cambio de valoración, generando un cambio en la valoración sugerida del contenedor.	Correcto
22.2	Ídem anterior	Ídem anterior	Correcto
23	Clase *Observation*	Prueba método *update*. Crea una práctica y un objetivo. Asigna el objetivo a la práctica. Luego modifica el objetivo para verificar que se regenere la valoración sugerida en la práctica.	Correcto
24	Clase *Observation*	Prueba método *notifyDelete*. Crea una práctica y un objetivo. Asigna el objetivo a la práctica. Luego invoca a *notifyDelete* en el objetivo para verificar que se regenere la valoración sugerida en la práctica.	Correcto
25	Clase *AppraisalManager*	Prueba método *createAppraisal*. Crea una evaluación y verifica si se ha creado correctamente.	Correcto
26.1	Clase *AppraisalManager*	Prueba método *initLevel*. Crea una evaluación, inicializa un nivel de madurez dentro de la misma y verifica si se ha inicializado correctamente.	Correcto

CP	Componente	Descripción	Resultado
26.2	Ídem anterior	Ídem anterior	Correcto
26.3	Ídem anterior	Ídem anterior	Correcto
27	Clase *AppraisalManager*	Prueba método *saveAppraisal*. Crea una evaluación, la completa la almacena.	Correcto
28	Clase *AppraisalManager*	Prueba método *openAppraisal*. Crea una evaluación y la completa. Lee una evaluación de archivo y la compara con la anterior.	Correcto
29	Clase *AssessmentManager*	Prueba método *setAssessment* para áreas de proceso. Crea una evaluación con un nivel de madurez y sus áreas de proceso. Setea las valoraciones de todas las áreas de proceso y verifica que se genere automáticamente la valoración del nivel de madurez.	Correcto
30	Clase *AssessmentManager*	Prueba método *setAssessment* para objetivos. Crea una evaluación con un nivel de madurez, sus áreas de proceso y objetivos. Setea las valoraciones de todos los objetivos de un área de proceso y verifica que se genere automáticamente la valoración del área de proceso.	Correcto
31	Clase *AssessmentManager*	Prueba método *setAssessment* para prácticas. Crea una evaluación con un nivel de madurez, sus áreas de proceso, objetivos, y prácticas. Setea las valoraciones de todas las prácticas de un objetivo y verifica que se genere automáticamente la valoración del objetivo.	Correcto
32	Clase *ObservationManager*	Prueba método *createObservation*. Crea una evaluación completa (incluyendo observaciones). Crea una nueva observación en la evaluación y verifica que se haya creado bien.	Correcto
33	Clase *ObservationManager*	Prueba método *updateObservation*. Crea una evaluación completa (incluyendo observaciones). Modifica una observación en la evaluación y verifica que se haya modificado bien.	Correcto
34	Clase *ObservationManager*	Prueba método *deleteObservation*. Crea una evaluación completa (incluyendo observaciones). Elimina una observación en la evaluación y verifica que se haya eliminado bien.	Correcto

Tabla 9.1: resultados de las pruebas unitarias y de integración.

9.4 Pruebas del sistema

Una vez finalizada la codificación correspondiente a cada una de las iteraciones del desarrollo, se procedió a la ejecución de las pruebas del sistema, de acuerdo a lo especificado en el apartado 8.3.2 (Pruebas del sistema) del capítulo 8.

En los apartados siguientes se muestran los resultados de la ejecución de las pruebas, condensados para todas las iteraciones de desarrollo. Algunos de estos casos de prueba fueron ejecutados durante la iteración 2 (Arquitectura) y otros durante la iteración 3 (Construcción).

9.4.1 Pruebas funcionales

Caso de uso: Administrar evaluaciones

La tabla 9.2 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba el inicio de una evaluación a nivel de prácticas con instancias	Correcto
1	2	Prueba el inicio de una evaluación a nivel de prácticas con instancias, dejando las instancias en blanco	Incorrecto. No aparece ningún mensaje y permite agregar la instancia sin ningún nombre. Luego, al evaluar un nivel de madurez, la instancia en blanco aparece en el árbol del modelo. **Severidad: Media**
1	3	Prueba el inicio de una evaluación a nivel de prácticas sin instancias	Correcto
1	4	Prueba el inicio de una evaluación a nivel de objetivos	Correcto
1	5	Prueba el inicio de una evaluación a nivel de áreas de proceso	Correcto
1	6	Prueba de inicio de evaluación con datos inválidos	Correcto
1	7	Prueba de inicio de evaluación cancelando la operación	Correcto
2	1	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	Correcto
2	2	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	Correcto
3	1	Prueba de almacenamiento de evaluación	Incorrecto. Al intentar verificar las instancias Proyecto A y Proyecto B no aparece la ventana de evaluación de prácticas. Se analizó este defecto y el mismo se repite efectuando los siguientes pasos: 1. Evaluar una instancia 2. Almacenar la evaluación 3. Abrir nuevamente la evaluación almacenada 4. Evaluar la misma instancia (esta vez no aparece la ventana de evaluación) Lo mismo ocurre para Prácticas, Objetivos, Áreas de proceso y Niveles de madurez. **Severidad: Alta**
3	2	Prueba de almacenamiento de evaluación cancelando la operación	Correcto
4	1	Prueba de recuperación de evaluación	Correcto

Esc	CP	Descripción	Resultado
4	2	Prueba de recuperación de evaluación cancelando la operación	Correcto
4	3	Prueba de recuperación de evaluación con archivo inexistente	Correcto
5	1	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	Correcto
5	2	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	Correcto

Tabla 9.2: resultados de las pruebas funcionales del sistema para el caso de uso "Administrar evaluaciones".

Caso de uso: Administrar observaciones

La tabla 9.3 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	CP	Descripción	Resultado
1	1	Prueba el listado de observaciones existentes cuando no hay observaciones	Correcto
2	1	Prueba de creación de observaciones	Correcto
2	2	Prueba de creación de observaciones sin datos	Correcto
3	1	Prueba de modificación de observación	Correcto
3	2	Prueba de modificación de observación sin alterar datos	Correcto
4	1	Prueba de modificación de observación con regeneración de valoración en prácticas vinculadas	Correcto
5	1	Prueba de eliminación de observación	Correcto
5	2	Prueba de eliminación de observación con regeneración de valoración en prácticas vinculadas	Correcto

Tabla 9.3: resultados de las pruebas funcionales del sistema para el caso de uso "Administrar observaciones".

Caso de uso: Evaluar niveles de madurez

La tabla 9.4 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	CP	Descripción	Resultado
1	1	Prueba el inicio de la evaluación de nivel de madurez	Correcto
1	2	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor Alcanzado	Correcto
1	3	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor No alcanzado	Correcto
1	4	Ídem	Correcto
1	5	Prueba de asignación de valoración al nivel evaluado	Correcto
2	1	Prueba de evaluación de nivel de madurez 3 sin evaluar niveles inferiores	Correcto
2	2	Prueba de evaluación de nivel de madurez 4 sin evaluar niveles inferiores	Correcto
2	3	Prueba de evaluación de nivel de madurez 5 sin evaluar niveles inferiores	Correcto
2	4	Prueba de evaluación de nivel de madurez con evaluación de nivel inferior	Correcto

Esc	CP	Descripción	Resultado
3	1	Prueba de evaluación de nivel 3 con alcance prácticas y nivel 2 alcanzado	Correcto
3	2	Prueba de evaluación de nivel 3 con alcance objetivos y nivel 2 alcanzado	Correcto
3	3	Prueba de evaluación de nivel 3 con alcance áreas de proceso y nivel 2 alcanzado	Correcto

Tabla 9.4: resultados de las pruebas funcionales del sistema para el caso de uso "Evaluar niveles de madurez".

Caso de uso: Evaluar áreas de proceso

La tabla 9.5 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	CP	Descripción	Resultado
1	1	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Satisfecho	Correcto
1	2	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Insatisfecho	Correcto
1	3	Prueba de asignación de valoración elegida al área de proceso evaluada	Correcto
1	4	Prueba de asignación de valoración elegida sin asignar valor	Correcto
2	1	Prueba de evaluación de área de proceso sin evaluar objetivos	Correcto
2	2	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Fuera de alcance	Correcto
2	3	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Sin puntaje	Correcto
3	1	Prueba de evaluación con alcance áreas de proceso	Correcto
3	2	Prueba de asignación de valoración elegida con alcance a nivel de áreas de proceso	Correcto

Tabla 9.5: resultados de las pruebas funcionales del sistema para el caso de uso "Evaluar áreas de proceso".

Caso de uso: Evaluar objetivos

La tabla 9.6 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	CP	Descripción	Resultado
1	1	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Satisfecho	Correcto
1	2	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho	Correcto

Esc	*CP*	*Descripción*	*Resultado*
1	3	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho por observaciones tipo Debilidad	Incorrecto. Al vincular las 3 observaciones a la práctica bajo evaluación se produce un error. Las observaciones 1 y 2 se vinculan bien, pero al vincular la observación 3, en la lista de observaciones vinculadas quedan 2-1-2-3 (se repite la 2 y se agrega al comienzo). Si se desvincula la 1ra aparición de la obs 2, la lista se convierte en 1-3-1-3 Si ahora se desvincula la 1ra aparicion de la obs 1, la lista se convierte en 3-3-3 Por último, si se desvincula la 1ra aparición de la obs 3, la lista queda con una única aparición de la obs 3, la cual ya no se puede desvincular **Severidad: Alta**
1	4	Prueba de asignación de valoración elegida al objetivo evaluado	Correcto
1	5	Prueba de asignación de valoración elegida sin asignar valor	Incorrecto. Deja asignar valoración nula al objetivo. **Severidad: Baja**
2	1	Prueba de evaluación de objetivo sin evaluar prácticas	Correcto
3	1	Prueba de evaluación con alcance objetivos	Correcto
3	2	Prueba de asignación de valoración elegida con alcance a nivel de objetivos	Correcto

Tabla 9.6: resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar objetivos”.

Caso de uso: Evaluar Prácticas

La tabla 9.7 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Completamente Implementada	Correcto
1	2	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Correcto

Esc	*CP*	*Descripción*	*Resultado*
1	3	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Parcialmente Implementada	Incorrecto. Al asignar valoración Parcialmente Implementada en la 1ra de las instancias, se marcó con una cruz roja todo el camino N2->AP01->OE1->P01 (No ocurre lo mismo con las valoraciones Completamente Implementada y Ampliamente Implementada) **Severidad: Media**
1	4	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada	Incorrecto. Al asignar valoración No Implementada en la 1ra de las instancias, se marcó con una cruz roja todo el camino N2->AP01->OE1->P01 (No ocurre lo mismo con las valoraciones Completamente Implementada y Ampliamente Implementada) **Severidad: Media**
1	5	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Correcto
1	6	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada o Parcialmente Implementada	Correcto
1	7	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada, Parcialmente Implementada o Ampliamente Implementada	Correcto
1	8	Prueba de asignación de valoración elegida a la práctica evaluada	Correcto
1	9	Prueba de asignación de valoración elegida sin asignar valor	Correcto
2	1	Prueba de evaluación de práctica sin evaluar instancias	Correcto
3	1	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Completamente Implementada	Correcto
3	2	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Ampliamente Implementada	Incorrecto. Se da el mismo defecto que en el caso de prueba 1.3 del caso de uso “Evaluar Objetivos” No obstante, la valoración sugerida es correcta. **Severidad: Baja**
3	3	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Parcialmente Implementada	Incorrecto. Se da el mismo defecto que en el caso de prueba 1.3 del caso de uso “Evaluar Objetivos” No obstante, la valoración sugerida es correcta. **Severidad: Baja**

Esc	*CP*	*Descripción*	*Resultado*
3	4	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor No Implementada	Correcto
3	5	Prueba de asignación de valoración elegida con alcance a nivel prácticas sin instancias	Correcto

Tabla 9.7: resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar prácticas”.

Caso de uso: Generar reporte

La tabla 9.8 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba la generación de un reporte para alcance a nivel de Áreas de proceso	Correcto
1	2	Prueba de campo Justificación grande (múltiples renglones)	Correcto
1	3	Prueba de campo Justificación grande (que exceda la longitud de un renglón)	Correcto
1	4	Prueba la generación de un reporte para alcance a nivel de Objetivos	Correcto
1	5	Prueba la generación de un reporte para alcance a nivel de Prácticas	Correcto
1	6	Prueba de generación de reporte a nivel de Prácticas con todos los niveles evaluados	Correcto

Tabla 9.8: resultados de las pruebas funcionales del sistema para el caso de uso “Generar reporte”.

9.4.2 Pruebas de rendimiento

La tabla 9.9 muestra los resultados de las pruebas de rendimiento del sistema.

CP	*Descripción*	*Resultado*
1	Prueba el inicio de una evaluación a nivel de Prácticas con instancias	Correcto
2	Prueba de evaluación de Nivel de madurez	Correcto
3	Prueba de inferencia de valoraciones a nivel de instancia	Correcto
4	Prueba de asignación de valoración elegida a nivel de instancia	Correcto
5	Prueba de inferencia de valoraciones a nivel conjunto de instancias	Correcto
6	Prueba de inferencia de valoraciones a nivel de Práctica sin instancias	Correcto
7	Prueba de asignación de valoración elegida a nivel de Práctica sin instancias	Correcto
8	Prueba de inferencia de valoraciones a nivel Objetivo	Correcto
9	Prueba de inferencia de valoraciones a nivel Área de proceso	Correcto
10	Prueba de inferencia de valoraciones a nivel Nivel de madurez	Correcto
11	Prueba de almacenamiento de evaluación en curso	Correcto
12	Prueba de apertura de evaluación almacenada	Correcto
13	Prueba de propagación de modificaciones en observaciones	Correcto
14	Prueba de generación de reporte	Correcto

Tabla 9.9: resultados de las pruebas de rendimiento del sistema.

9.4.3 Evaluación de los resultados

Las pruebas del sistema permitieron descubrir algunos defectos de la aplicación. La tabla 9.10 resume los defectos encontrados.

Nro de defecto	*Descripción*	*Severidad*	*Caso de prueba*
1	La aplicación permite iniciar una evaluación con alcance a nivel de prácticas y e instancias en blanco (sin nombre). Luego al evaluar los niveles de madurez, las instancias en blanco aparecen como hojas sin nombre colgando de las prácticas.	Media	Caso de uso “Administrar evaluaciones”, caso de prueba 1.2
2	Cuando se evalúa una Práctica, y a continuación se almacena y recupera la evaluación en curso, no se puede volver a evaluar la misma práctica. Lo mismo ocurre para Objetivos, Áreas de proceso y Niveles de madurez.	Alta	Caso de uso “Administrar evaluaciones”, caso de prueba 3.1
3	No se puede vincular más de 2 observaciones a una Práctica. Al vincular la observación nro 3, se produce un defecto que desordena la lista de observaciones vinculadas e impide desvincularlas.	Alta	Caso de uso “Evaluar objetivos”, caso de prueba 1.3
4	La aplicación permite asignar una valoración elegida nula en la ventana de Evaluación de Objetivos.	Baja	Caso de uso “Evaluar objetivos”, caso de prueba 1.5
5	Cuando se asigna valoración elegida “Parcialmente Implementada” a una práctica, se infiere la valoración del objetivo que la contiene (esto es incorrecto ya que debe inferirse cuando se terminan de asignar valoraciones a TODAS las prácticas)	Media	Caso de uso “Evaluar prácticas”, caso de prueba 1.3
6	Cuando se asigna valoración elegida “No Implementada” a una práctica, se infiere la valoración del objetivo que la contiene (esto es incorrecto ya que debe inferirse cuando se terminan de asignar valoraciones a TODAS las prácticas)	Media	Caso de uso “Evaluar prácticas”, caso de prueba 1.4

Tabla 9.10: resumen de defectos encontrados en las pruebas del sistema.

Dado que se encontraron defectos con severidad Alta y Media, el sistema no pasó el criterio de aceptación definido en el apartado 8.1.2 (Pruebas del sistema) del capítulo 8. Por tal motivo, se efectuó la corrección de todos los defectos encontrados y se repitieron nuevamente las pruebas. Luego de las correcciones no aparecieron nuevos defectos.

10. IMPLANTACIÓN Y ACEPTACIÓN DEL SISTEMA

En este capítulo se documentan los productos de salida del proceso IAS (Implantación y Aceptación del Sistema) de la metodología Métrica V3 [Métrica V3, 2000]. El mismo se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

10.1 Incorporación del sistema al entorno de operación

El entorno de operación que utilizado para la implantación y aceptación del sistema fue el siguiente:

- Computadora Personal con las siguientes características:
  - Procesador AMD XP 2200, de 1.8 GHz de velocidad
  - 512 MB de memoria RAM
- Ambiente de ejecución J2SE versión 1.4.2

Con el entorno de operación disponible, se procedió a la instalación del sistema de acuerdo a la documentación provista en el apartado 13.3 (Manual del usuario) del capítulo 13 (Anexos). La instalación fue exitosa y no se registraron incidentes.

10.2 Pruebas de implantación y aceptación

Una vez finalizada la instalación, se procedió a la ejecución de las pruebas de implantación y aceptación, de acuerdo a lo especificado en el apartado 8.3.3 (Pruebas de implantación y aceptación) del capítulo 8.

Observación: dado que las prueba de aceptación son básicamente las mismas que las de sistema, los resultados de las mismas fueron correctos al primer intento. Esto se debe a que las pruebas de sistema se habían ejecutado previamente durante las iteraciones 2 (Arquitectura) y 3 (Construcción), con lo cual ya se habían corregido todos los defectos encontrados.

10.2.1 Pruebas funcionales

Caso de uso: Administrar evaluaciones

La tabla 10.1 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba el inicio de una evaluación a nivel de prácticas con instancias	Correcto
1	2	Prueba el inicio de una evaluación a nivel de prácticas con instancias, dejando las instancias en blanco	Correcto
1	3	Prueba el inicio de una evaluación a nivel de prácticas sin instancias	Correcto
1	4	Prueba el inicio de una evaluación a nivel de objetivos	Correcto
1	5	Prueba el inicio de una evaluación a nivel de áreas de proceso	Correcto
1	6	Prueba de inicio de evaluación con datos inválidos	Correcto
1	7	Prueba de inicio de evaluación cancelando la operación	Correcto
2	1	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	Correcto
2	2	Prueba el inicio de una evaluación cuando existe otra evaluación en curso	Correcto
3	1	Prueba de almacenamiento de evaluación	Correcto
3	2	Prueba de almacenamiento de evaluación cancelando la operación	Correcto
4	1	Prueba de recuperación de evaluación	Correcto
4	2	Prueba de recuperación de evaluación cancelando la operación	Correcto
4	3	Prueba de recuperación de evaluación con archivo inexistente	Correcto
5	1	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	Correcto
5	2	Prueba la recuperación de una evaluación cuando existe otra evaluación en curso	Correcto

Tabla 10.1. Resultados de las pruebas funcionales del sistema para el caso de uso “Administrar evaluaciones”.

Caso de uso: Administrar observaciones

La tabla 10.2 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba el listado de observaciones existentes cuando no hay observaciones	Correcto
2	1	Prueba de creación de observaciones	Correcto
2	2	Prueba de creación de observaciones sin datos	Correcto
3	1	Prueba de modificación de observación	Correcto
3	2	Prueba de modificación de observación sin alterar datos	Correcto
4	1	Prueba de modificación de observación con regeneración de valoración en prácticas vinculadas	Correcto
5	1	Prueba de eliminación de observación	Correcto
5	2	Prueba de eliminación de observación con regeneración de valoración en prácticas vinculadas	Correcto

Tabla 10.2. Resultados de las pruebas funcionales del sistema para el caso de uso “Administrar observaciones”.

Caso de uso: Evaluar niveles de madurez

La tabla 10.3 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba el inicio de la evaluación de nivel de madurez	Correcto
1	2	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor Alcanzado	Correcto
1	3	Prueba de evaluación de nivel de madurez con inferencia de valoración con valor No alcanzado	Correcto
1	4	Ídem	Correcto
1	5	Prueba de asignación de valoración al nivel evaluado	Correcto
2	1	Prueba de evaluación de nivel de madurez 3 sin evaluar niveles inferiores	Correcto
2	2	Prueba de evaluación de nivel de madurez 4 sin evaluar niveles inferiores	Correcto
2	3	Prueba de evaluación de nivel de madurez 5 sin evaluar niveles inferiores	Correcto
2	4	Prueba de evaluación de nivel de madurez con evaluación de nivel inferior	Correcto
3	1	Prueba de evaluación de nivel 3 con alcance prácticas y nivel 2 alcanzado	Correcto
3	2	Prueba de evaluación de nivel 3 con alcance objetivos y nivel 2 alcanzado	Correcto
3	3	Prueba de evaluación de nivel 3 con alcance áreas de proceso y nivel 2 alcanzado	Correcto

Tabla 10.3. Resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar niveles de madurez”.

Caso de uso: Evaluar áreas de proceso

La tabla 10.4 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Satisfecho	Correcto
1	2	Prueba de evaluación de área de proceso con inferencia de valoración sugerida valor Insatisfecho	Correcto
1	3	Prueba de asignación de valoración elegida al área de proceso evaluada	Correcto
1	4	Prueba de asignación de valoración elegida sin asignar valor	Correcto
2	1	Prueba de evaluación de área de proceso sin evaluar objetivos	Correcto
2	2	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Fuera de alcance	Correcto
2	3	Prueba de evaluación de área de proceso sin evaluar objetivos, con área Sin puntaje	Correcto
3	1	Prueba de evaluación con alcance áreas de proceso	Correcto
3	2	Prueba de asignación de valoración elegida con alcance a nivel de áreas de proceso	Correcto

Tabla 10.4. Resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar áreas de proceso”.

Caso de uso: Evaluar objetivos

La tabla 10.5 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Satisfecho	Correcto
1	2	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho	Correcto
1	3	Prueba de evaluación de objetivo con inferencia de valoración sugerida valor Insatisfecho por observaciones tipo Debilidad	Correcto
1	4	Prueba de asignación de valoración elegida al objetivo evaluado	Correcto
1	5	Prueba de asignación de valoración elegida sin asignar valor	Correcto
2	1	Prueba de evaluación de objetivo sin evaluar prácticas	Correcto
3	1	Prueba de evaluación con alcance objetivos	Correcto
3	2	Prueba de asignación de valoración elegida con alcance a nivel de objetivos	Correcto

Tabla 10.5. Resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar objetivos”.

Caso de uso: Evaluar Prácticas

La tabla 10.6 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Completamente Implementada	Correcto
1	2	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Correcto
1	3	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Parcialmente Implementada	Correcto
1	4	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada	Correcto
1	5	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada	Correcto
1	6	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor Ampliamente Implementada o Parcialmente Implementada	Correcto
1	7	Prueba de evaluación de práctica con inferencia de valoración sugerida a partir de instancias, valor No Implementada, Parcialmente Implementada o Ampliamente Implementada	Correcto
1	8	Prueba de asignación de valoración elegida a la práctica evaluada	Correcto
1	9	Prueba de asignación de valoración elegida sin asignar valor	Correcto
2	1	Prueba de evaluación de práctica sin evaluar instancias	Correcto
3	1	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Completamente Implementada	Correcto
3	2	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Ampliamente Implementada	Correcto
3	3	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor Parcialmente Implementada	Correcto
3	4	Prueba de evaluación sin instancias con inferencia de valoración sugerida valor No Implementada	Correcto
3	5	Prueba de asignación de valoración elegida con alcance a nivel prácticas sin instancias	Correcto

Tabla 10.6. Resultados de las pruebas funcionales del sistema para el caso de uso “Evaluar prácticas”.

Caso de uso: Generar reporte

La tabla 10.7 muestra los resultados de las pruebas funcionales para este caso de uso.

Esc	*CP*	*Descripción*	*Resultado*
1	1	Prueba la generación de un reporte para alcance a nivel de Áreas de proceso	Correcto
1	2	Prueba de campo Justificación grande (múltiples renglones)	Correcto
1	3	Prueba de campo Justificación grande (que exceda la longitud de un renglón)	Correcto
1	4	Prueba la generación de un reporte para alcance a nivel de Objetivos	Correcto
1	5	Prueba la generación de un reporte para alcance a nivel de Prácticas	Correcto
1	6	Prueba de generación de reporte a nivel de Prácticas con todos los niveles evaluados	Correcto

Tabla 10.7. Resultados de las pruebas funcionales del sistema para el caso de uso “Generar reporte”.

10.2.2 Pruebas de rendimiento

La tabla 10.8 muestra los resultados de las pruebas de rendimiento del sistema.

CP	*Descripción*	*Resultado*
1	Prueba el inicio de una evaluación a nivel de Prácticas con instancias	Correcto
2	Prueba de evaluación de Nivel de madurez	Correcto
3	Prueba de inferencia de valoraciones a nivel de instancia	Correcto
4	Prueba de asignación de valoración elegida a nivel de instancia	Correcto
5	Prueba de inferencia de valoraciones a nivel conjunto de instancias	Correcto
6	Prueba de inferencia de valoraciones a nivel de Práctica sin instancias	Correcto
7	Prueba de asignación de valoración elegida a nivel de Práctica sin instancias	Correcto
8	Prueba de inferencia de valoraciones a nivel Objetivo	Correcto
9	Prueba de inferencia de valoraciones a nivel Área de proceso	Correcto
10	Prueba de inferencia de valoraciones a nivel Nivel de madurez	Correcto
11	Prueba de almacenamiento de evaluación en curso	Correcto
12	Prueba de apertura de evaluación almacenada	Correcto
13	Prueba de propagación de modificaciones en observaciones	Correcto
14	Prueba de generación de reporte	Correcto

Tabla 10.8. Resultados de las pruebas de rendimiento del sistema.

10.2.3 Evaluación de los resultados

El sistema pasó exitosamente las pruebas de implantación y aceptación, sin producirse ningún incidente. El usuario confirmó la aceptación del sistema.

11. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

En este capítulo se vuelcan las conclusiones obtenidas luego de finalizado el trabajo de tesis, y las futuras líneas de trabajo a seguir por aquellos interesados en el tema.

11.1 Conclusiones sobre la metodología

El desarrollo de la tesis se efectuó siguiendo paso a paso la metodología Métrica V3. La misma significó una ayuda para el desarrollo, posibilitando un avance ordenado a través de sus procesos y actividades. Asimismo, los documentos y modelos propuestos en la metodología facilitaron la documentación de todas las actividades realizadas.

Como crítica a Métrica V3, se puede resaltar que carece de guías detalladas para realización de algunas actividades, principalmente las relacionadas con la construcción de algunos modelos UML. Esto hace que se dificulte el modelado en orientación a objetos utilizando esta metodología, obligando a recurrir a otras fuentes en algunos casos, como por ejemplo al Proceso Unificado [Booch & Jacobson & Rumbaugh, 2000].

Otro aspecto criticable de Métrica V3 es su carencia de plantillas para la generación de documentos y modelos.

No obstante, el balance general en el uso de la metodología resulta altamente positivo.

11.2 Conclusiones sobre la tecnología

El entorno tecnológico seleccionado para la herramienta resultó adecuado y no se presentaron mayores inconvenientes durante el desarrollo. Esto permite concluir que la tecnología Java se encuentra en un estado de madurez suficiente como para ser utilizada en la construcción de este tipo de herramientas.

11.3 Conclusiones sobre las herramientas

Las herramientas utilizadas durante el desarrollo mostraron ser altamente efectivas, aumentando notablemente la productividad y posibilitando el ordenamiento de todos los productos intermedios del desarrollo.

A continuación se efectúa un breve análisis de cada una de ellas:

Enterprise Architect

Se utilizó para generar y mantener toda la documentación del sistema, desde los requerimientos iniciales hasta el diseño de las clases, manteniendo la trazabilidad entre todos los elementos.

Una vez que se tuvo el diseño, se generó el código fuente, el cual posteriormente fue utilizado para sincronizar nuevamente el diseño (ingeniería de ida y vuelta entre el diseño y el código fuente).

Como balance general se puede decir que la herramienta cubre la mayoría de las necesidades del desarrollo de software en lo que respecta al modelado de requerimientos, análisis, y diseño. Asimismo, se puede destacar su amplia capacidad de generación de reportes, los cuales permitieron construir gran parte de la documentación de la tesis.

Eclipse

Se utilizó para el desarrollo del código fuente y la automatización de las pruebas unitarias y de integración.

Luego de generado el código fuente inicial desde *Enterprise Architect*, se programaron todos los métodos de las clases dentro de esta herramienta. A medida que se programaba, se generaban las pruebas unitarios y de integración utilizando la integración que provee el entorno con la herramienta JUnit [JUnit, 2004]. Finalmente, las pruebas se ejecutaban dentro del entorno, corrigiendo inmediatamente los defectos que surgían.

Como crítica a esta herramienta se puede mencionar que carece de un entorno para la construcción de interfaces gráficas de usuario, lo que obliga a utilizar otras herramientas para esa tarea o a codificar manualmente las interfaces.

Como balance general se puede decir que la herramienta cubre la mayoría de las necesidades asociadas a un entorno de programación. Su interfaz es altamente amigable, y posee características que agilizan notablemente el desarrollo de software.

NetBeans

Se utilizó solamente para la construcción de las interfaces gráficas. Luego de construidas las interfaces, se incorporó el código generado por la herramienta al entorno *Eclipse*, donde se integró con el resto del código de la aplicación.

Como balance general se puede decir que la herramienta es bastante pobre si se la compara con *Eclipse*, en cuanto a las características relacionadas con la programación. Sin embargo cuenta con un entorno para la construcción de interfaces de usuario bastante potente, motivo por el cual se la adoptó.

ClearCase LT

Se utilizó como herramienta de gestión de configuraciones durante todo el proyecto.

Como balance general se puede decir que la herramienta cumplió satisfactoriamente con todos los requerimientos establecidos en el apartado 5.2 (Gestión de la configuración) del capítulo 5. Se destaca su facilidad de uso, su simplicidad en la instalación y configuración, y sus amplias capacidades de visualización y gestión de versiones y líneas base.

11.4 Conclusiones sobre el trabajo

Como primera conclusión sobre el trabajo efectuado, se puede mencionar que la estimación inicial fue bastante precisa.

Durante el anteproyecto, el esfuerzo estimado fue de 900 horas/hombre, luego al comienzo del trabajo, se refinó la estimación arrojando 700 horas/hombre. Con una dedicación de aproximadamente 20 hs/semana, el trabajo debía completarse en aproximadamente 35 semanas.

Analizando retrospectivamente, se verificó que el trabajo de tesis comenzó a mediados de junio del 2003 y finalizó a principios de febrero del 2004, arrojando un total de aproximadamente 33 semanas (2 semanas menos de lo previsto en la estimación refinada).

Como segunda conclusión sobre el trabajo, se destaca que se alcanzaron exitosamente todos los objetivos establecidos para la herramienta. El producto obtenido cuenta con todas las características que se relevaron como deseables al inicio de la tesis, resultando efectivamente en una herramienta de ayuda para la evaluación del modelo.

11.5 Futuras líneas de trabajo

En primera instancia, la herramienta obtenida será utilizada de manera experimental para actividades de evaluación informal del modelo CMMI-SW dentro de la consultora Snoop Consulting (lugar de trabajo del tesista). Esto brindará una realimentación propia del uso en entornos reales, que seguramente dará origen a nuevas propuestas de ampliaciones y/o modificaciones.

Independientemente de esto, dentro de las posibles líneas de trabajo en la herramienta, se pueden destacar las siguientes:

- Agregar otras facilidades de reportes, sobre todo gráficos.
- Agregar soporte para la evaluación del modelo CMMI-SW en su representación *Continua* (actualmente sólo soporta la representación *Por niveles*).
- Agregar soporte para la evaluación de todos los demás modelos de la familia CMMI.
- Adaptar la capa de presentación de la herramienta, para que pueda funcionar en modo web (esto implica replantear las *views* del modelo de diseño, reutilzando todos los demás paquetes).

12. BIBLIOGRAFÍA Y GLOSARIO

12.1 Bibliografía

Appraisal Wizard, 2003. *Formal or informal appraisal tool*, Integrated System Diagnostics Incorporated. Demo disponible en el sitio de la empresa, http://www.isd-inc.com, página vigente al 30/01/2004.

Booch & Jacobson & Rumbaugh, 1998. *The Unified Modelling Language Reference Manual*. 576 páginas. Editorial Addison-Wesley. ISBN 020130998X.

Booch & Jacobson & Rumbaugh, 2000. *El Proceso Unificado de Desarrollo de Software*. 438 páginas. Editorial Addison-Wesley. ISBN 8478290362.

Chrissis & Konrad & Shrum, 2003. *CMMI. Guidelines for Process Integration and Product Improvement*. 688 páginas. Editorial Addison-Wesley. ISBN 0321154967.

ClearCase LT, 2004. *Rational ClearCase LT*, IBM-Rational. Disponible en http://www-306.ibm.com/software/awdtools/clearcase/cclt/, página vigente al 30/01/2004.

CMM, 1991. *Capability Maturity Model for Software*. Disponible en http://www.sei.cmu.edu/cmm/, página vigente al 30/01/2004.

CMM-Quest, 2001. *Self assessment tool*, HM&S IT-Consulting GmbH. Demo disponible en el sitio de la empresa, http://www.cmm-quest.com/, página vigente al 30/01/2004.

CMMI, 2002. *Capability Maturity Model Integration*. Disponible en http://www.sei.cmu.edu/cmmi/cmmi.html, página vigente al 30/01/2004.

CMMI Adoption, 2003. *CMMI Adoption Information*. Disponible en http://www.sei.cmu.edu/cmmi/adoption/adoption.html, página vigente al 29/09/2003.

CMMI-SW, 2002. *Software Engineering Capability Maturity Model Integration*. Disponible en http://www.sei.cmu.edu/cmmi/models/models.html, página vigente al 29/01/2004.

CMMIS, 2002. *CMMI Models*. Disponibles en http://www.sei.cmu.edu/cmmi/models/models.html, página vigente al 30/01/2004.

CMMS, 2003. *Capability Maturity Models*. Disponibles en http://www.sei.cmu.edu/cmm/cmms/cmms.html, página vigente al 30/01/2004.

EA, 2004. *Enterprise Architect 3.60 User Guide*, SparxSystems. Disponible en http://www.sparxsystems.com.au/EAUserGuide/index.html, página vigente al 30/01/2004.

Eclipse, 2004. *Entorno Integrado de Desarrollo*, proyecto OpenSource impulsado por IBM. Disponible en http://www.eclipse.org, página vigente al 06/02/2004.

Hunt, J. 2001. *You've got the model-view-controller*. JayDee Technology Limited. Disponible en http://www.jaydeetechnology.co.uk/planetjava/tutorials/swing/Model-View-Controller.PDF, página vigente al 05/02/2004.

Hunt, J. 2001. *The Hierarchical MVC*. JayDee Technology Limited. Disponible en http://www.jaydeetechnology.co.uk/planetjava/tutorials/swing/hmvc.PDF, página vigente al 05/02/2004.

IEEE 830, 1993. *Recommended Practice for Software Requirements Specifications*, Software Engineering Standards Committee of the IEEE Computer Society: New York, NY, 1993.

IME Toolkit, 2003. *Interim Maturity Evaluation Toolkit*, Management Information Systems. Disponible en http://www.man-info-systems.com/IMEtoolkit.htm, página vigente al 30/01/2004.

J2SE, 2004. *Java 2 Standard Edition*, Sun Microsystems. Disponible en http://java.sun.com/j2se, página vigente al 02/02/2004.

JasperReports, 2004. *Java Reports, Report-generating tool*, proyecto OpenSource. Disponible en http://jasperreports.sourceforge.net/, página vigente al 05/02/2004.

JavaHelp, 2004. *Java Help System,* Sun Microsystems. Disponible en http://java.sun.com/products/javahelp/index.jsp, página vigente al 05/02/2004.

JavaBeans, 2004. *JavaBeans Technology,* Sun Microsystems. Disponible en http://java.sun.com/products/javabeans/, página vigente al 05/02/2004.

JUnit, 2004. *JUnit, Testing Resources for Extreme Programming,* proyecto OpenSource. Disponible en http://www.junit.org, página vigente al 05/02/2004.

Métrica V3, 2000. *Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información*. Ministerio de Administraciones Públicas Español. Disponible en http://www.csi.map.es/csi/metrica3/index.html, página vigente al 29/01/2004.

Motorola, 2003. *Entrevistas realizadas por el tesista a empleados de Motorola Argentina*. Agosto y septiembre, entrevistas orales.

Peralta, M.L. 2004. *Estimación del esfuerzo basada en Casos de Uso*. Artículo escrito como parte de los estudios de master en CAPIS, ITBA, diciembre de 2001. Disponible en la biblioteca de CAPIS.

Paulk, Mark C. & Curtis, Bill & Chrissis, Mary Beth & Weber, Charles V. 1993. *Capability Maturity Model, Version 1.1*. IEEE Software, Vol. 10, No. 4, pp. 18-27.

Ribu, K. 2001. *Estimating Object-Oriented Software Projects with Use Cases*. Tesis de master. Universidad de Oslo. Disponible en: http://heim.ifi.uio.no/~kribu/oppgave.pdf

SCAMPI, 2001. *Standard CMMI Appraisal Method for Process Improvement*. Disponible en http://www.sei.cmu.edu/publications/documents/01.reports/01hb001.html, página vigente al 29/01/2004.

SEIR, 2003. *Compiled list of Organizations who have Publicly Announced their Maturity Levels after having an Appraisal Performed*. Software Engineering Information Repository. Disponible en http://seir.sei.cmu.edu/pml, página vigente al 30/01/2004.

SEI, 2003. *Software Engineering Institute*. Disponible en http://www.sei.cmu.edu, página vigente al 29/01/2004.

SW-CMM Migration, 2001. *Migration from the SW-CMM to CMMI*. Disponible en http://www.sei.cmu.edu/cmmi/adoption/migration.html, página vigente al 29/01/2004.

The Standish Group, 2003. Página principal de la empresa. Disponible en http://www.standishgroup.com, página vigente al 30/01/2004.

12.2 Glosario

ASI: proceso de Análisis del Sistema de Información (metodología Métrica V3).

CAPIS: Centro de Actualización Permanente en Ingeniería de Software.

CAT: herramienta objeto de la tesis, significa *CMMI-SW Appraisal Tool*.

CMMI: Integración del Modelo de Madurez de Capacidad (*Capability and Maturity Model Integration*).

CMMI-SW: modelo específico de CMMI para la disciplina de Ingeniería de Software.

CMMI-SE/SW/IPPD/SS: modelo de CMMI que abarca las disciplinas de ingeniería de sistemas, ingeniería de software, desarrollo integrado de procesos y productos, y relación con los proveedores.

CSI: proceso de Construcción del Sistema de Información (metodología Métrica V3).

DSI: proceso de Diseño del Sistema de Información (metodología Métrica V3).

EVS: proceso de Estudio de Viabilidad del Sistema (metodología Métrica V3).

IAS: proceso de Implantación y Aceptación del Sistema (metodología Métrica V3).

ITBA: Instituto Tecnológico de Buenos Aires.

Métrica V3: metodología para el desarrollo de Sistemas de información.

SCAMPI: método de evaluación estándar de CMMI para la mejora de procesos (*Standard CMMI Appraisal Method for Process Improvement*).

SEI: Instituto de Ingeniería de Software (*Software Engineering Institute*) de la universidad Carnegie Mellon (EEUU). Autor de los modelos CMMI y del método SCAMPI.

OpenSource: movimiento mundial que pregona la idea del software libre, compartido y mejorado públicamente a través de la colaboración de cualquier desarrollador interesado.

13. ANEXOS

13.1 Dossier de aseguramiento de la calidad

En esta sección se documentan los productos de salida de las actividades correspondientes a la Interfaz "Aseguramiento de la calidad" de la metodología Métrica V3 [Métrica V3, 2000]. La misma se construyó en paralelo con la ejecución de cada una de las actividades y tareas de la metodología, documentando sus resultados en los distintos apartados.

13.1.1 Revisiones sobre los productos de análisis

13.1.1.1 Revisión de requisitos

Luego de finalizada la actividad ASI9 (Análisis de consistencia), se efectuó una revisión del Catálogo de requisitos. La misma arrojó como resultado que los requisitos se encuentran especificados de una manera adecuada. Tanto el tesista como el usuario estuvieron de acuerdo con los requisitos establecidos para el sistema.

13.1.1.2 Revisión de la consistencia entre productos

Luego de finalizada la actividad ASI9 (Análisis de consistencia), se efectuó una revisión del análisis llevado a cabo. La misma arrojó como resultado que el análisis de la consistencia entre los productos del Análisis del sistema fue adecuado. Los modelos obtenidos son correctos y trazables entre sí.

13.1.1.3 Revisión del plan de pruebas

Luego de finalizada la actividad ASI10 (Especificación del plan de pruebas), se efectuó una revisión del mismo.

Durante la revisión se verificó que se hayan definido los niveles de prueba a utilizar, y para cada nivel los perfiles involucrados, la planificación temporal, los criterios de aceptación, la generación y mantenimiento de casos de prueba, las técnicas de evaluación, y los entregables resultantes de las pruebas.

El resultado de la revisión arrojó que el plan de pruebas es adecuado.

13.1.2 Revisiones sobre los productos de diseño

13.1.2.1 Revisión de la verificación de la arquitectura del sistema

Luego de finalizada la actividad DSI7 (Verificación y aceptación de la arquitectura del sistema), se efectuó una revisión del análisis llevado a cabo. La misma arrojó como resultado que el análisis de la consistencia entre los productos del Diseño del sistema fue adecuado. Los modelos obtenidos son correctos y trazables entre sí.

13.1.2.2 Revisión del diseño de las pruebas

Luego de finalizada la actividad DSI10 (Especificación técnica del plan de pruebas), se efectuó una revisión de los casos de prueba diseñados para los distintos niveles de prueba.

Durante la revisión se verificó que los diseños efectuados cumplen con los criterios establecidos en el plan de pruebas, en cuando a las verificaciones a efectuar, los casos de prueba asociados a cada verificación, el registro de los resultados, y la detección de defectos.

La revisión arrojó como resultado que las pruebas diseñadas son adecuadas.

13.1.2.3 Revisión del plan de pruebas

Luego de finalizada la actividad DSI10 (Especificación técnica del plan de pruebas), se efectuó una revisión del Plan de pruebas.

Durante la revisión se verificó que se hayan definido los casos de prueba correspondientes a los niveles de prueba definidos en el plan, y que dichos casos de prueba ejerciten la mayor cantidad de escenarios posibles de manera de optimizar la detección de defectos.

El resultado de la revisión arrojó que el plan de pruebas es adecuado.

13.1.3 Revisiones sobre los productos de construcción

13.1.3.1 Revisión de la realización de las pruebas unitarias y de integración

Luego de finalizadas la actividades CSI2 (Generación del código de los componentes y procedimientos), CSI3 (Ejecución de las pruebas unitarias) y CSI4 (Ejecución de las pruebas de integración), las cuales se efectuaron en paralelo, se llevó a cabo una revisión de la realización de las pruebas unitarias y de integración.

Durante la revisión se verificó que se hayan ejecutado todos los casos de prueba definidos en el Plan de pruebas, y que se hayan corregido todos los defectos encontrados.

El resultado de la revisión arrojó que las pruebas unitarias y de integración se efectuaron de manera adecuada.

13.1.3.2 Revisión de la realización de las pruebas del sistema

Luego de finalizada la actividad CSI5 (Ejecución de las pruebas del sistema), se llevó a cabo una revisión de la realización de dichas pruebas.

Durante la revisión se verificó que se hayan ejecutado todos los casos de prueba definidos, y que se hayan evaluado y documentado los resultados de las pruebas.

Durante las pruebas se encontraron defectos, los cuales fueron corregidos luego de finalizadas las mismas. Por tal motivo, se repitió nuevamente el proceso de pruebas, y se volvió a verificar que el mismo sea adecuado.

El resultado de la revisión arrojó que las pruebas del sistema se efectuaron de manera adecuada.

13.1.3.3 Revisión de los manuales de usuario

Luego de finalizada la actividad CSI6 (Elaboración de los manuales de usuario), se llevó a cabo una revisión de la documentación generada.

Durante la revisión se verificó que la documentación de usuario contemple la instalación y configuración del sistema, y que brinde una guía de uso clara y concisa.

El resultado de la revisión arrojó que la documentación de usuario es adecuada.

13.1.4 Revisiones sobre los productos de implantación y aceptación

13.1.4.1 Revisión de la realización de las pruebas de aceptación

Luego de finalizada la actividad IAS6 (Pruebas de aceptación del sistema), se llevó a cabo una revisión de la realización de dichas pruebas.

Durante la revisión se verificó que se hayan ejecutado todos los casos de prueba definidos, y que se hayan evaluado y documentado los resultados de las pruebas.

El resultado de la revisión arrojó que las pruebas de aceptación del sistema se efectuaron de manera adecuada.

13.1.4.2 Registro de la aprobación de las pruebas de aceptación por el usuario

Luego de finalizada la actividad IAS6 (Pruebas de aceptación del sistema) y la revisión mencionada en el apartado anterior, el usuario confirmó su aceptación del sistema.

La aceptación se produjo por parte de la M. Ing. Paola Britos, en su carácter de usuario del sistema, el día 9 de febrero de 2004.

13.2 Glosario de términos correspondientes al análisis

El glosario de términos contiene todos los términos utilizados en el negocio bajo análisis con sus respectivas definiciones.

13.2.1 Área de proceso

Un área de proceso agrupa un conjunto de prácticas relacionadas que cuando son implementadas de manera colectiva, satisfacen un conjunto de objetivos considerados importantes para el aporte de mejoras significativas en esa área.

13.2.2 Artefactos directos

Son los elementos tangibles que resultan directamente de la implementación de una práctica del modelo CMMI-SW. Ejemplos de estos elementos son: documentos, entregables, materiales de entrenamiento, etc.

13.2.3 Artefactos indirectos

Son elementos que surgen como consecuencia de la implementación de una práctica del modelo CMMI-SW, pero que no constituyen en sí el propósito de la práctica. Ejemplos de estos elementos son: minutas de reunión, revisión de resultados, reportes de estado, mediciones de desempeño, etc.

13.2.4 Características comunes

Las características comunes proveen una forma de organizar las prácticas genéricas dentro de cada área de proceso. Son componentes del modelo que no se evalúan de ninguna manera, y su único propósito es el de proveer una forma de agrupamiento para la presentación de las prácticas genéricas.

Las características comunes son:

- Compromiso de realización
- Capacidad de realización
- Dirigiendo implementación
- Verificando implementación

13.2.5 Nivel de madurez

El nivel de madurez de una organización provee una manera de predecir el desempeño futuro de una organización en una disciplina específica o en un conjunto de disciplinas. Cada nivel de madurez consiste de un conjunto predefinido de áreas de proceso. Para la evaluación del nivel de madurez alcanzado por una organización, se evalúa la satisfacción de las áreas de proceso que conforman el nivel analizado.

La estructura general de cada uno de los niveles de madurez se muestra en la figura 13.1.

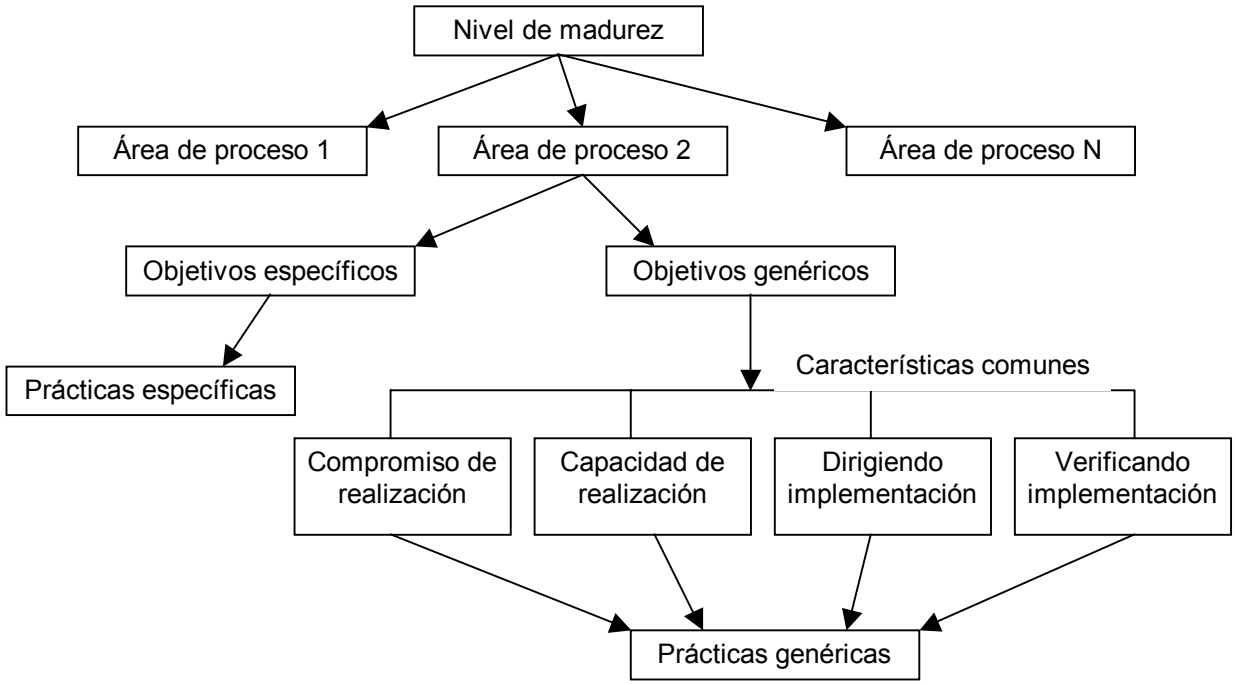

Figura 13.1. Estructura de un nivel de madurez.

Cada *nivel de madurez* agrupa un conjunto de *áreas de proceso*. Dentro de las áreas de proceso hay *objetivos específicos* y *objetivos genéricos*. Los objetivos específicos se alcanzan mediante *prácticas específicas*, y los objetivos genéricos mediante *prácticas genéricas*. Las prácticas genéricas se encuentran organizadas por *características comunes*.

Dentro del modelo CMMI-SW, existen los siguientes niveles de madurez:

- *Nivel 1: Inicial*
  En este nivel, los procesos son usualmente ad hoc y caóticos. El éxito depende de los individuos y no del uso de procesos probadamente exitosos.

Generalmente se exceden los presupuestos y los calendarios de los proyectos.

- *Nivel 2: Gestionado*
  En este nivel, la organización ha alcanzado todos los objetivos específicos y genéricos de las áreas de proceso vinculadas al nivel 2 del modelo. Los requerimientos son gestionados y los procesos son planificados, llevados a cabo, medidos y controlados.

- *Nivel 3: Definido*
  En este nivel, la organización ha alcanzado todos los objetivos específicos y genéricos de las áreas de proceso vinculadas a los niveles 2 y 3 del modelo. Los procesos están bien caracterizados y entendidos, y se encuentran descriptos en estándares, procedimientos, herramientas y métodos.

- *Nivel 4: Gestionado cuantitativamente*
  En este nivel, la organización ha alcanzado todos los objetivos específicos de las áreas de proceso vinculadas a los niveles 2, 3 y 4 del modelo; y los objetivos genéricos de los niveles 2 y 3. El desempeño y la calidad de los procesos se mide cuantitativamente y se analiza estadísticamente. Las causas de variación se identifican de manera de prevenir futuras desviaciones.

- *Nivel 5: Optimizado*
  En este nivel, la organización ha alcanzado todos los objetivos específicos de las áreas de proceso vinculadas a los niveles 2, 3, 4 y 5 del modelo; y los objetivos genéricos de los niveles 2 y 3. Los procesos son continuamente mejorados en base al entendimiento cuantitativo de las causas comunes de variación de los mismos.

13.2.6 Objetivos

Los objetivos son las metas deben alcanzarse para satisfacer las áreas de proceso. Se utilizan en las evaluaciones para determinar si un área de proceso es satisfecha o no.

Se dividen en:

- *Objetivos específicos*
  Son objetivos que aplican a una única área de proceso. Cada área de proceso tiene asociados un conjunto de objetivos específicos para ella.

- *Objetivos genéricos*

Son objetivos que aplican a múltiples áreas de proceso, a diferencia de los objetivos específicos que aplican a una única área. En la representación por niveles del modelo CMMI-SW, cada área de proceso tiene un único objetivo genérico.

13.2.7 Observaciones

Son elementos que se recopilan durante el análisis de una organización o proyecto.

Las observaciones resaltan las fortalezas y debilidades encontradas en la organización con respecto a la implementación de las prácticas de CMMI-SW.

Generalmente se utilizan para documentar las debilidades encontradas en la organización.

13.2.8 Prácticas

Son las actividades llevadas a cabo en una organización o proyecto, que permiten determinar la satisfacción de los objetivos. Se utilizan en las evaluaciones para determinar si un objetivo es satisfecho o no.

Se dividen en:

- *Prácticas específicas*
  Una práctica específica es una actividad considerada como importante para alcanzar el objetivo específico al cual está asociada. La implementación de las prácticas específicas implica la satisfacción de los objetivos específicos, lo que a su vez implica la satisfacción de las áreas de proceso.

- *Prácticas genéricas*
  Las prácticas genéricas proveen la institucionalización necesaria para asegurar que los procesos asociados con el área de proceso sean efectivos, repetibles y duraderos.

13.2.9 Valoración del nivel de madurez

Es el resultado final del proceso de evaluación.

Las valoraciones posibles son:

- Nivel alcanzado
- Nivel no alcanzado

13.2.10 Valoraciones de áreas de proceso

A cada área de proceso se le asigna una valoración de acuerdo al método SCAMPI.

Las valoraciones posibles son:

- Satisfecha
- Insatisfecha
- No aplicable
- Sin puntaje

13.2.11 Valoraciones de objetivos

A cada objetivo se le asigna una valoración de acuerdo al método SCAMPI.

Las valoraciones posibles son:

- Satisfecho
- Insatisfecho

13.2.12 Valoraciones de prácticas

A cada práctica se le asigna una valoración de acuerdo al método SCAMPI.

Las valoraciones posibles son:

- CI: completamente implementada
- AI: ampliamente implementada
- PI: parcialmente implementada
- NI: no implementada

13.3 Manual del usuario

El sistema es un asistente para la evaluación del modelo CMMI-SW en su representación por niveles, basado en el método de evaluación formal SCAMPI.

Las características principales con que cuenta son las siguientes:

- Guías online para la evaluación de las diferentes partes del modelo CMMI.
- Generación automática de valoraciones sugeridas basadas en cálculos efectuados sobre las valoraciones asignadas por el usuario.
- Facilidades para almacenamiento y recuperación de evaluaciones.
- Generación de reportes con los resultados de las evaluaciones.

13.3.1 Instalación y ejecución de la aplicación

La distribución del sistema consiste de un único archivo llamado *cat.zip*. Este archivo contiene todos los ejecutables y librerías necesarios para el funcionamiento del sistema.

La instalación se puede resumir en los siguientes pasos:

1. Instalar la plataforma J2SE 1.4.2 o superior
2. Descomprimir el archivo *cat.zip* en cualquier directorio del disco rígido

Una vez completada la instalación, la aplicación se puede arrancar ejecutando el siguiente comando (en una ventana de comandos del sistema operativo):

java –jar cat.jar

13.3.2 Configuración

Descomprimiendo el archivo *cat.jar*, se pueden ver entre otros los elementos que se muestran en el listado 13.1.

```
controllers/
messages/
META-INF/
models/
mvc/
persistence/
views/
  |--- guides/
  |       |--- es/
  |--- helps/
          |--- es/
AllTests.class
config.properties
Main.class
messages_es.properties
```

Listado 13.1: contenido del archivo de cat.jar.

El archivo *config.properties* contiene los parámetros de configuración del sistema. A continuación se muestra el contenido de dicho archivo:

```
#
# Archivo de configuración
#
locale = es
```

la variable *locale* permite definir el idioma a utilizar (en el caso del ejemplo es Español, codificado como *es*). El valor de dicha variable se utiliza dentro de la aplicación para recuperar el archivo de mensajes y etiquetas (*messages_<locale>.properties*), las guías online (archivos html dentro del directorio *views/guides/<locale>*), y las ayudas (archivos html dentro del directorio *views/help/<locale>*).

El archivo *messages_<locale>.properties* contiene la definición de todos los mensajes y etiquetas utilizados en la aplicación.

El directorio *views/guides/<locale>* contiene todas las guías que se muestran en las ventanas de evaluación de los distintos componentes del modelo.

El directorio *views/helps/<locale>* contiene las ayudas de la aplicación.

De esta manera, para dar soporte a otro idioma bastaría con editar el archivo *config.properties* indicando la extensión del nuevo idioma, y por supuesto proveer el archivo de mensajes y los directorios de guías y ayudas correspondientes.

Por ejemplo, para que la aplicación funcione en inglés se debe proveer lo siguiente:

1) Archivo *config.properties* con la siguiente configuración

```
locale = en
```

2) Archivo *messages_en.properties* con todas las etiquetas y mensajes en inglés

3) Directorio *views/guides/en* con las guías en inglés

4) Directorio *views/helps/en* con las ayudas en inglés

Finalmente, cambiando el valor de la variable *locale* en el archivo *config.properties* la aplicación podría funcionar alternativamente en español o en inglés.

13.3.3 Guía de uso

Esta sección muestra detalladamente la operación del sistema desde el punto de vista del usuario. La aproximación utilizada para la documentación consiste en partir del caso típico de evaluación, y destacar como casos particulares las variantes que se dan al cambiar el alcance de las evaluaciones.

13.3.3.1 Ventana principal

Al ejecutar la aplicación se abre la ventana principal de la misma, dando la posibilidad de iniciar una nueva evaluación, abrir una evaluación almacenada, consultar la ayuda, o salir del sistema. La figura 13.2 muestra el aspecto de esta ventana.

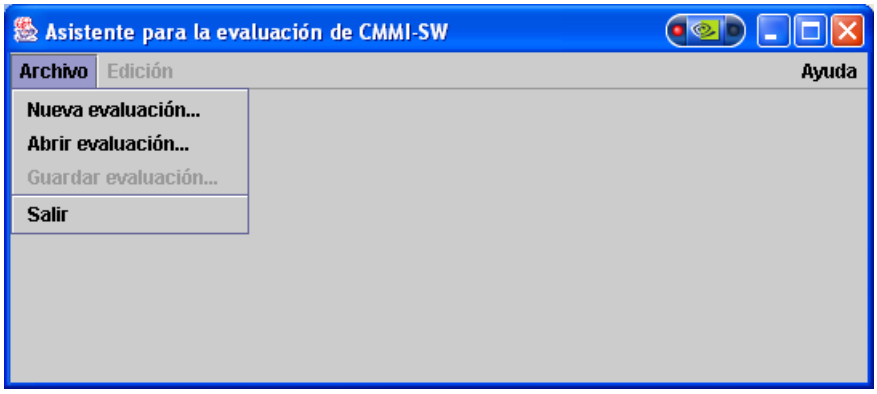

Figura 13.2. Ventana principal de la aplicación mostrando las opciones del menú Archivo..

13.3.3.2 Nueva evaluación

Al seleccionar la opción "Nueva evaluación" del menú Archivo, se abre la ventana de inicio de evaluación. La misma puede observarse en la figura 13.3.

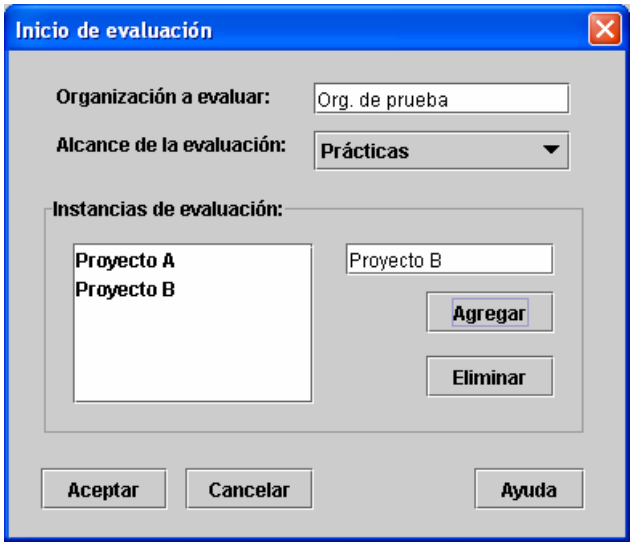

Figura 13.3. Ventana de inicio de evaluación..

En esta ventana, se pueden especificar los siguientes parámetros:

Organización a evaluar: permite definir el nombre de la organización bajo análisis.

Alcance de la evaluación: permite definir el alcance que se dará a la evaluación. Los valores posibles son los siguientes:

- **Prácticas**: el nivel de detalle en la aplicación del método de evaluación SCAMPI llega hasta las Prácticas específicas y genéricas del modelo CMMI-SW.
- **Objetivos**: el nivel de detalle en la aplicación del método de evaluación SCAMPI llega hasta los Objetivos específicos y genéricos del modelo CMMI-SW.
- **Áreas de proceso**: el nivel de detalle en la aplicación del método de evaluación SCAMPI llega hasta las Áreas de proceso del modelo CMMI-SW.

Para el alcance a nivel de Prácticas, se habilita la posibilidad de ingresar **Instancias de evaluación**. Las Instancias de evaluación son proyectos específicos dentro de la organización a evaluar.

De acuerdo al método SCAMPI, las prácticas pueden evaluarse a nivel de instancia (proyecto) y a nivel de conjunto (de instancias). Alternativamente, también se puede evaluar las prácticas sin definir instancias, con lo cual se evalúa la práctica en sí misma, a nivel de organización, sin evaluar proyectos específicos.

La aplicación se encuentra preparada para ambos tipos de evaluación.

Una vez que se indican los parámetros, el botón Aceptar inicia la nueva evaluación. La ventana principal de la aplicación se divide en un panel izquierdo donde se muestra el árbol con la representación por niveles del modelo CMMI-SW, y un panel derecho con un área para las ventanas de evaluación. La figura 13.4 muestra la ventana principal luego de iniciada una evaluación.

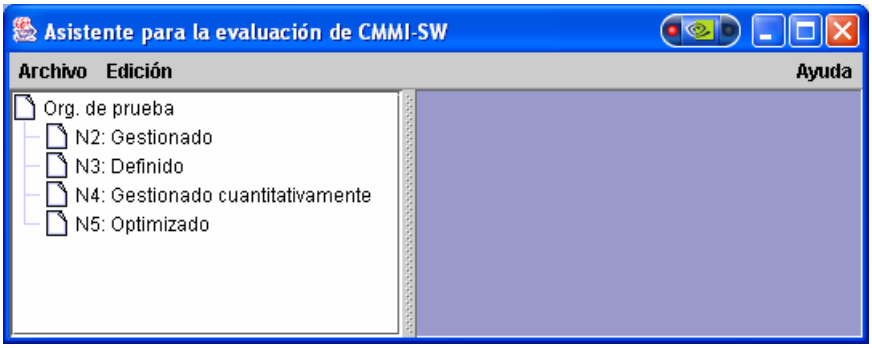

Figura 13.4. Ventana principal luego del inicio de una evaluación. Cuando se inicia una nueva evaluación, el árbol con la representación del modelo muestra solamente los niveles de madurez.

13.3.3.3 Apertura de evaluación existente

Al seleccionar la opción "Abrir evaluación" del menú Archivo, se abre la ventana de apertura de archivos. La figura 13.5 muestra el aspecto de la misma.

En esta ventana se puede navegar por la estructura de directorios y archivos del disco rígido, seleccionando un archivo a abrir mediante el parámetro **Nombre de archivo**. Una vez que se selecciona un archivo, el botón Abrir lo lee e inicializa la aplicación con los datos almacenados en el mismo.

La ventana principal de la aplicación se divide en un panel izquierdo donde se muestra el árbol con la representación por niveles del modelo CMMI-SW, y un panel derecho con un área para las ventanas de evaluación. La figura 13.6 muestra la ventana principal luego de abrirse una evaluación desde archivo.

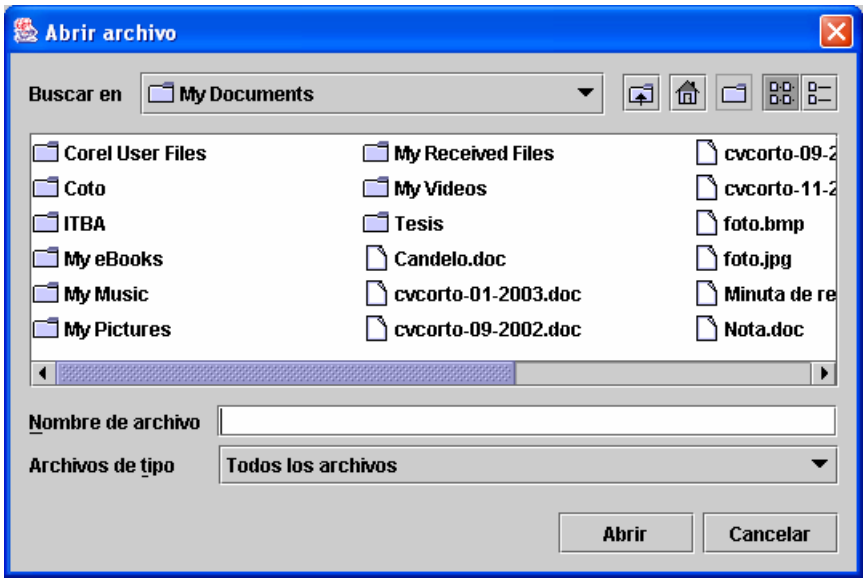

Figura 13.5: Ventana de apertura de archivos.

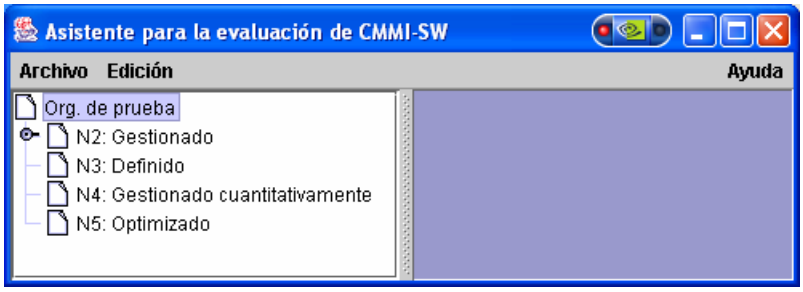

Figura 13.6. Ventana principal luego de la apertura de una evaluación almacenada.

13.3.3.4 Almacenamiento de una evaluación en curso

Al seleccionar la opción “Guardar evaluación” del menú Archivo, se abre la ventana de almacenamiento de archivos. La figura 13.7 muestra el aspecto de la misma.

En esta ventana se puede navegar por la estructura de directorios y archivos del disco rígido, seleccionando un archivo donde almacenar la evaluación mediante el parámetro **Nombre de archivo**. Una vez que se selecciona un archivo, el botón Guardar almacena la evaluación en curso en el mismo.

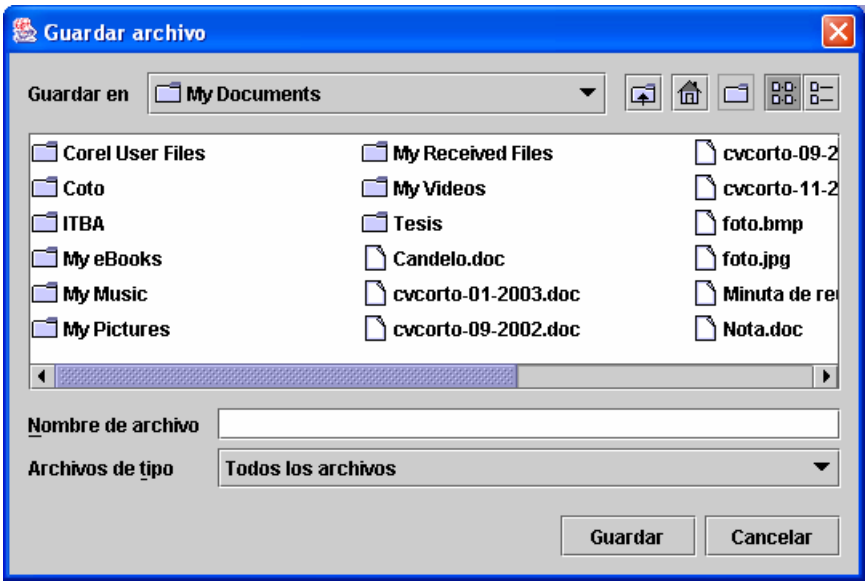

Figura 13.7: Ventana de almacenamiento de archivos.

13.3.3.5 Salir del sistema

Al seleccionar la opción "Salir" del menú Archivo, la aplicación se cierra. En caso de existir una evaluación en curso, antes de cerrarse, la aplicación pregunta al usuario si desea almacenar la evaluación, presentando una ventana como la que se muestra en la figura 13.8.

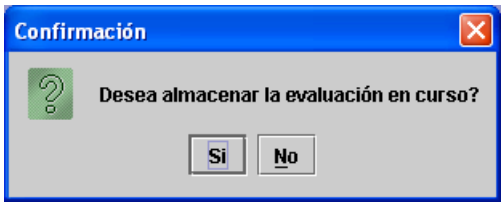

Figura 13.8. Ventana de solicitud de confirmación para almacenar la evaluación en curso.

En caso de seleccionar la opción **Si**, se abre la ventana de almacenamiento de evaluación tal como se indica en la sección "Almacenamiento de una evaluación en curso".

En caso de seleccionar la opción **No**, se cierra la aplicación descartando la evaluación en curso.

13.3.3.6 Administración de observaciones

Al seleccionar la opción “Observaciones” del menú Edición, se abre la ventana de Administración de observaciones. La figura 13.9 muestra el aspecto de la misma.

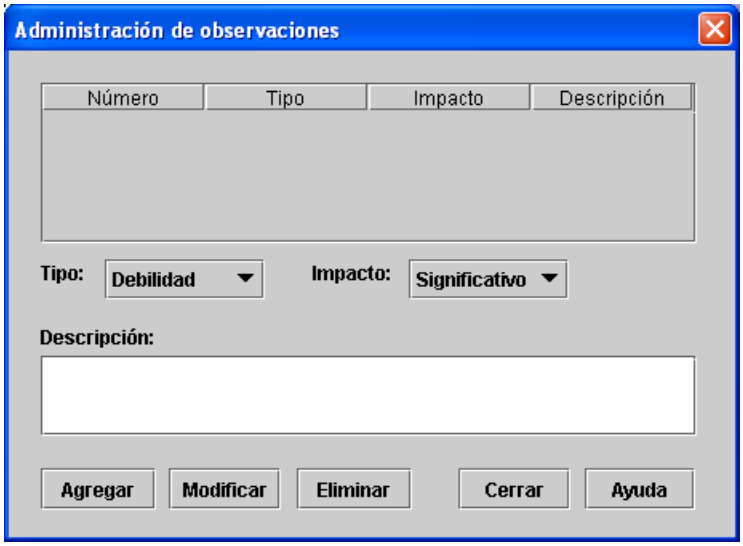

Figura 13.9. Ventana de Administración de observaciones.

En esta ventana se pueden administrar las observaciones que se identifiquen durante la evaluación del modelo CMMI-SW. Cada observación se encuentra caracterizada por los siguientes parámetros:

- **Número**: identificador de la observación. Es un número secuencial asignado automáticamente por el sistema.

- **Tipo**: tipo de observación. Los valores posibles son “Debilidad” o “Fortaleza”.

- **Impacto**: impacto asociado a la observación. Los valores posibles son “Significativo” y “No significativo”.

- **Descripción**: texto descriptivo asociado a la observación.

Las observaciones definidas se vinculan posteriormente a las Prácticas durante la evaluación de las mismas.

Para definir una nueva observación, se deben indicar los parámetros **Tipo**, **Impacto**, y **Descripción**, y a continuación presionar el botón Agregar. El sistema define la nueva observación y la muestra en la lista de observaciones existentes en la parte

superior de la ventana. La figura 13.10 muestra la ventana luego de agregar una nueva observación.

Figura 13.10. Ventana de Administración de observaciones luego de definir una nueva observación.

Para modificar una observación existente, se debe seleccionar la misma de la lista de observaciones existentes. Esto hace que los campos **Tipo**, **Impacto**, y **Descripción** se llenen con los valores de la observación seleccionada. Modificando cualquiera de los campos y presionando el botón Modificar la observación seleccionada queda modificada.

Para eliminar una observación existente, se debe seleccionar la misma de la lista de observaciones existentes, y a continuación presionar el botón Eliminar.

Al modificar o eliminar observaciones existentes, el sistema recalcula las valoraciones de las Prácticas que tengan vinculaciones con las observaciones modificadas o eliminadas.

13.3.3.7 Evaluación de los componentes del modelo

Al seleccionar algún elemento sobre el árbol con la representación por niveles del modelo CMMI-SW (panel izquierdo de la ventana principal) y a continuación la opción “Evaluar” del menú Edición, se abre la ventana de evaluación del elemento seleccionado.

De manera equivalente, seleccionando un elemento en el árbol y presionando el botón derecho del mouse, aparece un menú flotante con la opción “Evaluar”.

La figura 13.11 muestra los elementos susceptibles de ser evaluados.

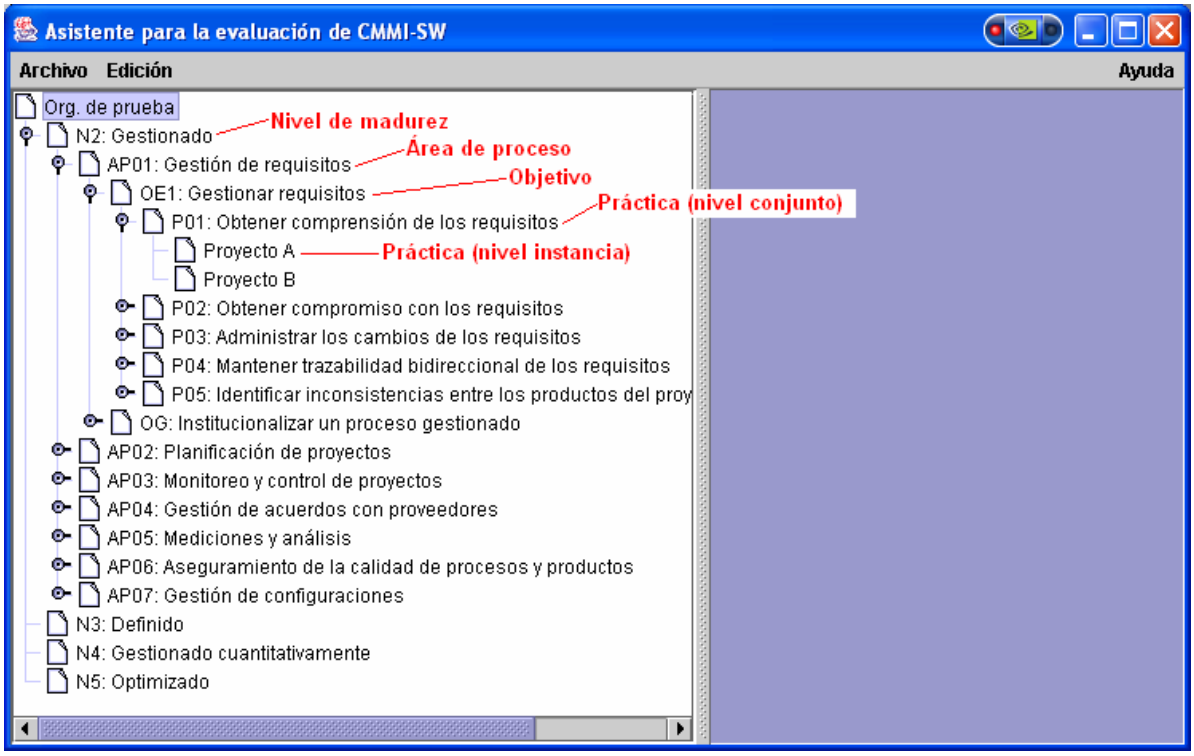

Figura 13.11: Ventana principal de la aplicación mostrando los distintos elementos del modelo CMMI-SW que pueden evaluarse en la herramienta.

Para efectuar una evaluación completa del modelo, el método de evaluación SCAMPI comienza seleccionando el nivel de madurez a evaluar. Una vez seleccionado el nivel de madurez, el método evalúa los elementos de menor nivel en el modelo y utiliza las valoraciones de esos elementos para sugerir las valoraciones de los elementos del nivel inmediato superior.

Por ejemplo, en una evaluación del Nivel 2 del modelo CMMI-SW con alcance a nivel de Prácticas y dos instancias específicas (Proyecto A y Proyecto B), como se muestra en la figura anterior, el método comienza evaluando las instancias Proyecto A y Proyecto B de la práctica P01 (ver figura 13.11). Las valoraciones asignadas a esas instancias se utilizan para generar una valoración sugerida para la práctica P01.

Luego, cuando se terminan de asignar valoraciones a las prácticas P01 a P05, se genera una valoración sugerida para el objetivo OE1.

De la misma manera, cuando se terminan de asignar valoraciones a los objetivos OE1 y OG, se genera una valoración sugerida para el área de proceso AP01.

Finalmente, cuando se termina de asignar valoraciones a las áreas de proceso AP01 a AP07, se genera una valoración sugerida para el nivel de madurez N2.

Evaluación de Nivel de madurez

Para evaluar un nivel de madurez, se debe seleccionar el mismo en el árbol de navegación del modelo, y a continuación la opción “Evaluar” del menú Edición, o bien, botón derecho del mouse y opción “Evaluar”. La figura 13.12 muestra esta última opción.

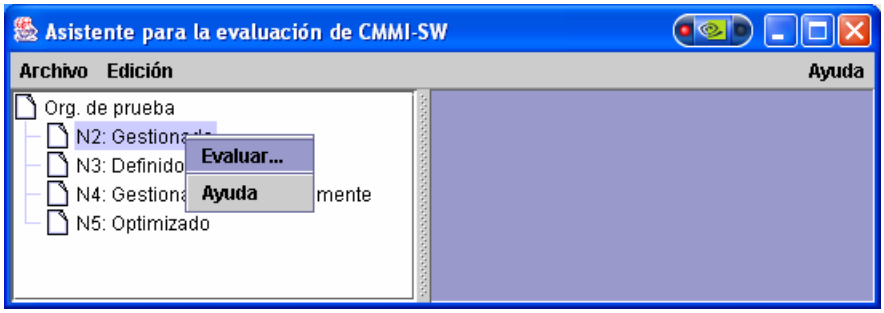

Figura 13.12. Ventana principal de la aplicación mostrando la selección del Nivel 2 para comenzar su evaluación.

Al iniciar la evaluación del nivel de madurez, el árbol del modelo despliega las áreas de proceso asociadas al nivel seleccionado, y se abre la ventana de Evaluación de nivel de madurez, como se muestra en la figura 13.13.

La evaluación del nivel de madurez consiste en ir evaluando todos los elementos del modelo, comenzando por los de nivel inferior, hasta llegar a las áreas de proceso. Una vez que se evalúan todas las áreas de proceso, la ventana de Evaluación de nivel de madurez muestra la valoración concluida para el nivel evaluado. La figura 13.14 muestra la ventana principal luego de evaluadas todas las áreas de proceso.

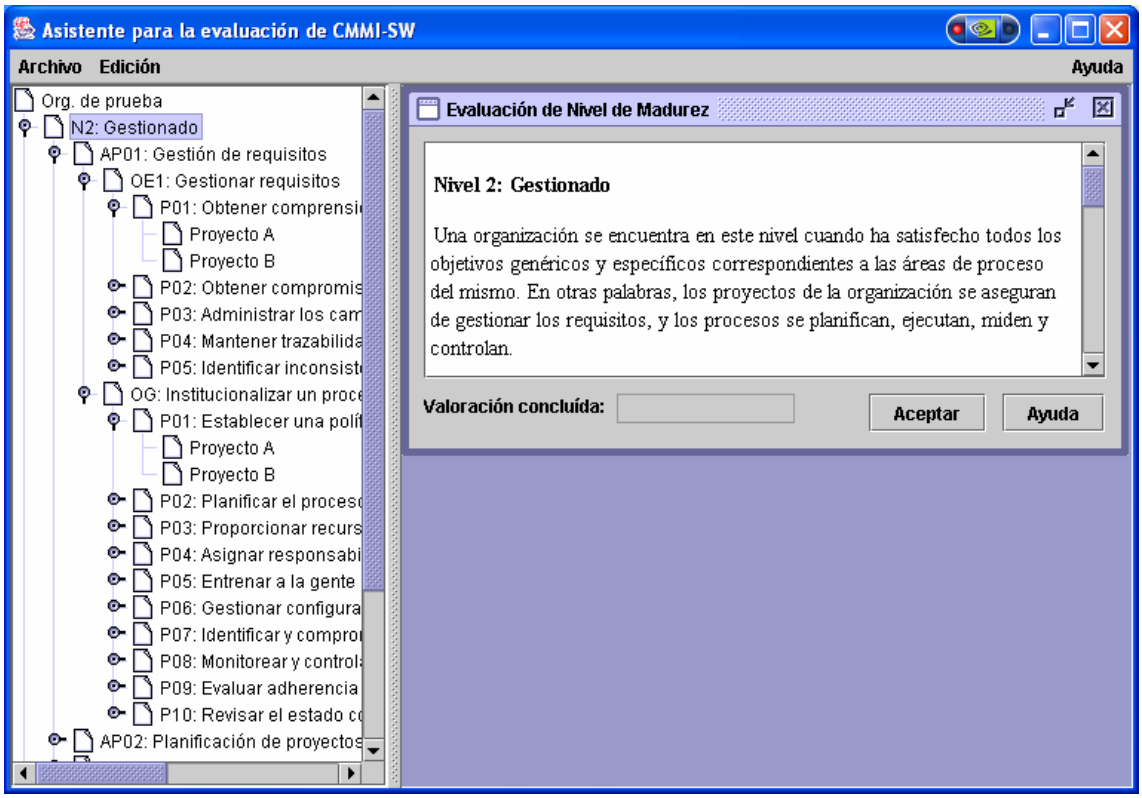

Figura 13.13. Ventana principal de la aplicación mostrando la evaluación del Nivel 2. En el panel izquierdo (árbol de navegación del modelo) pueden verse las áreas de proceso, los objetivos y las prácticas asociados al nivel de madurez. En el panel derecho puede verse la ventana de Evaluación de nivel de madurez mostrando la guía online para la evaluación del nivel, y un área para la valoración concluida.

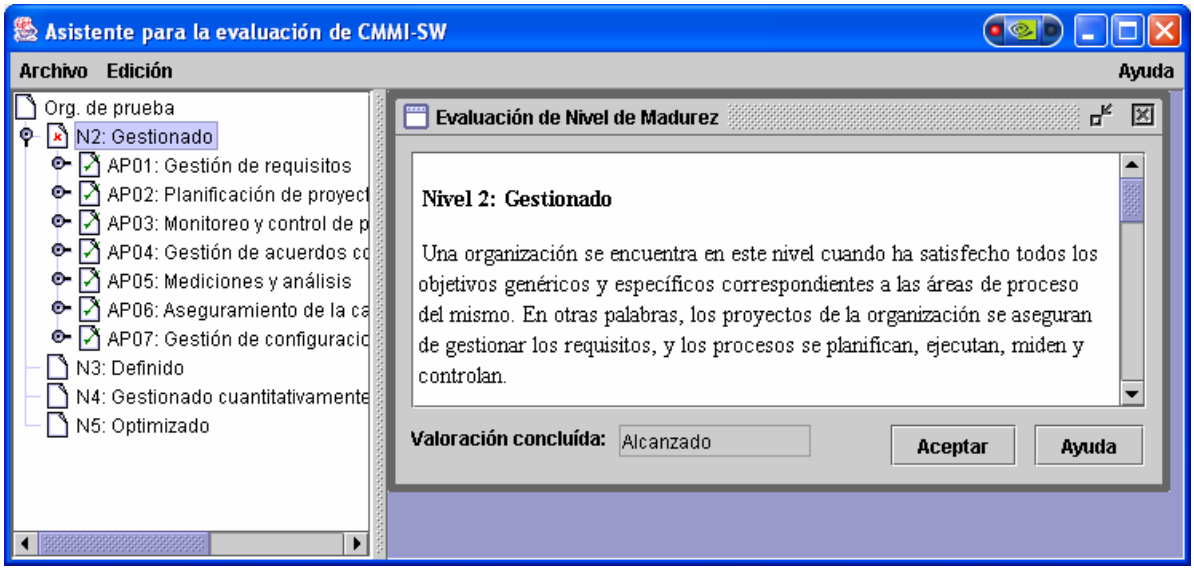

Figura 13.14. Ventana principal de la aplicación mostrando la evaluación del Nivel 2. En el panel izquierdo (árbol de navegación del modelo) pueden verse las áreas de proceso ya evaluadas y el nivel N2 marcado con una cruz roja, lo que significa que hay una valoración concluida para el mismo que aún no ha sido aceptada.

Finalmente, al presionar el botón Aceptar en la ventana de Evaluación de nivel de madurez, se acepta la valoración concluida y se cierra la ventana. El nivel de madurez queda evaluado, y en el árbol de navegación del modelo la cruz roja se cambia por una marca en verde. La figura 13.15 muestra la ventana principal luego de aceptada la valoración concluida para el nivel 2.

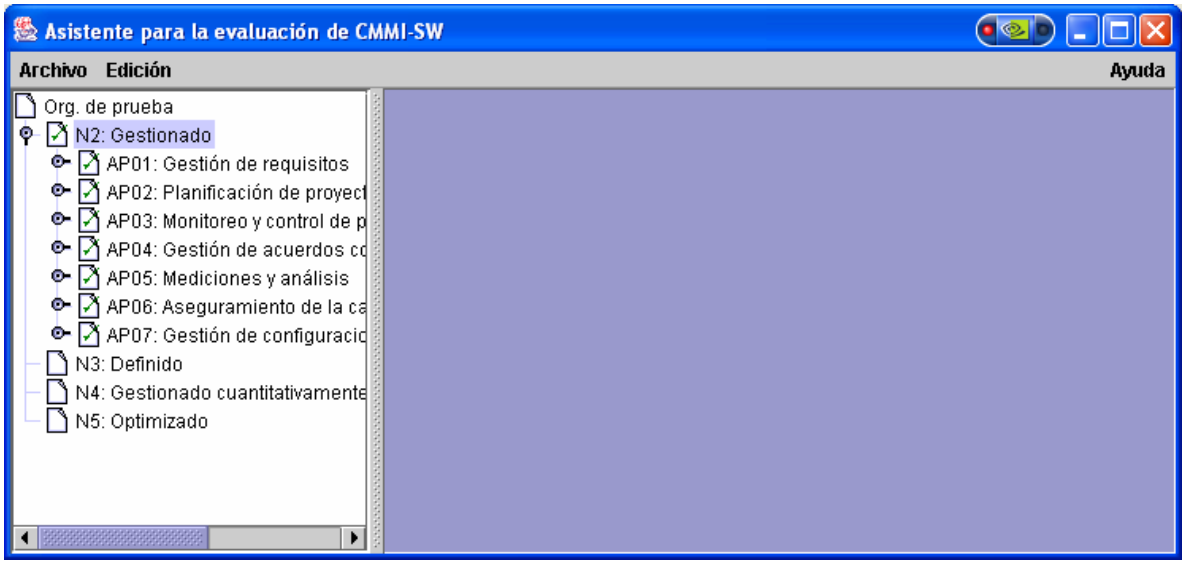

Figura 13.15: Ventana principal de la aplicación mostrando el Nivel 2 evaluado.

Caso particular: Evaluación de Nivel de madurez sin evaluar niveles inferiores

Un caso particular de la evaluación de niveles de madurez se da cuando se desea evaluar un nivel sin haber evaluado con valoración "Alcanzado" todos los niveles inferiores. En este caso, la aplicación muestra la advertencia que mostrada en la figura 13.16.

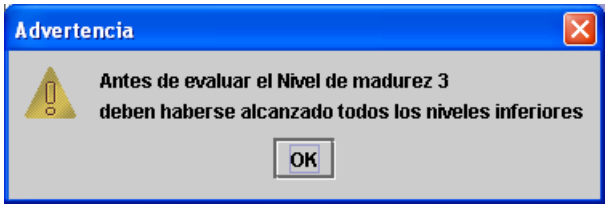

Figura 13.16. Ventana de advertencia al intentar evaluar el nivel de madurez 3 sin haber evaluado el nivel 2.

Aceptando la advertencia, se abre la ventana de Evaluación de niveles de madurez y se seguir adelante con la evaluación.

Caso particular: Evaluación de Nivel 3

Otro caso particular de la evaluación de niveles de madurez se da cuando se desea evaluar el nivel 3. De acuerdo a la documentación del modelo CMMI-SW, al evaluar el nivel 3 se cambian los objetivos genéricos utilizados para la evaluación del nivel 2, lo cual obliga a reevaluar dichos objetivos.

El sistema contempla esa situación de la siguiente manera. Cuando el alcance de la evaluación llega hasta el nivel de las prácticas y ya se ha evaluado el nivel 2, al iniciar la evaluación del nivel 3 se regeneran los objetivos genéricos del nivel 2, sin valoración asignada, de manera de repetir la evaluación.

Las figuras 13.17, 13.18 y 13.19 muestran las pantallas del sistema en esta situación.

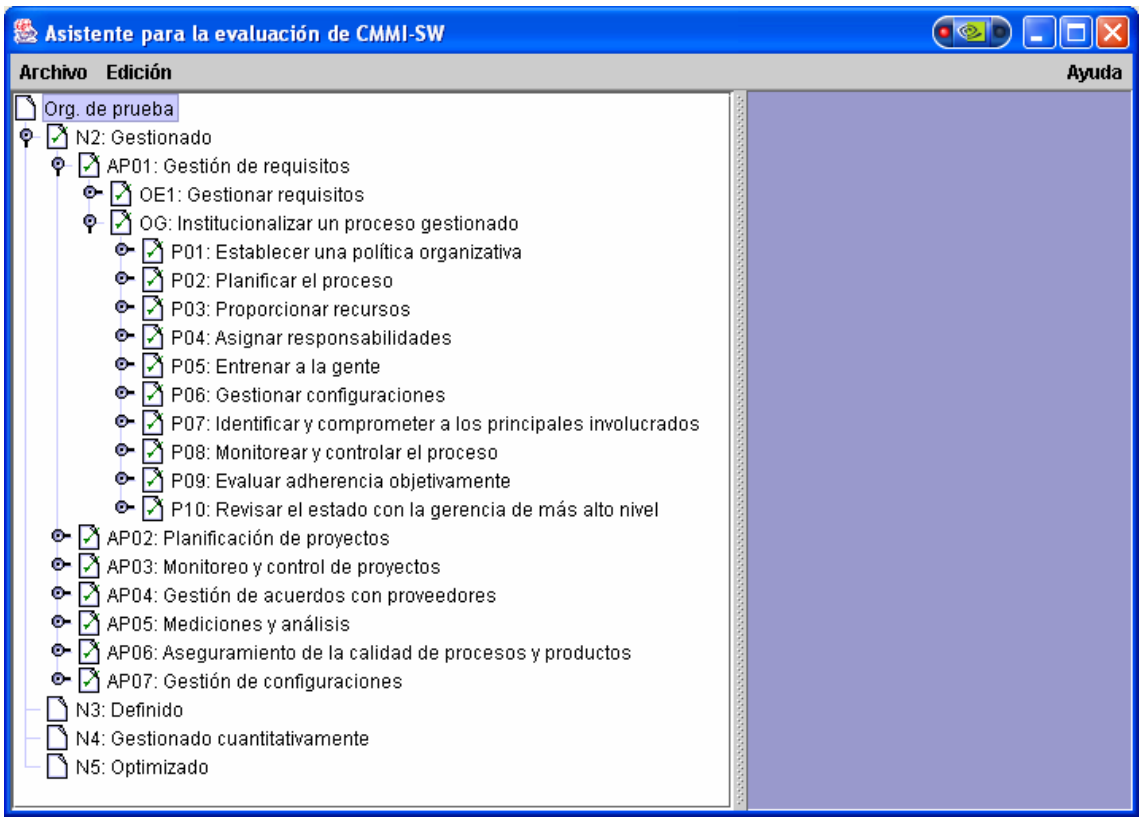

Figura 13.17. Ventana principal de la aplicación luego de evaluado el nivel 2. En el árbol de navegación del modelo puede verse un Objetivo Genérico (OG: Institucionalizar un proceso gestionado) con sus 10 prácticas asociadas.

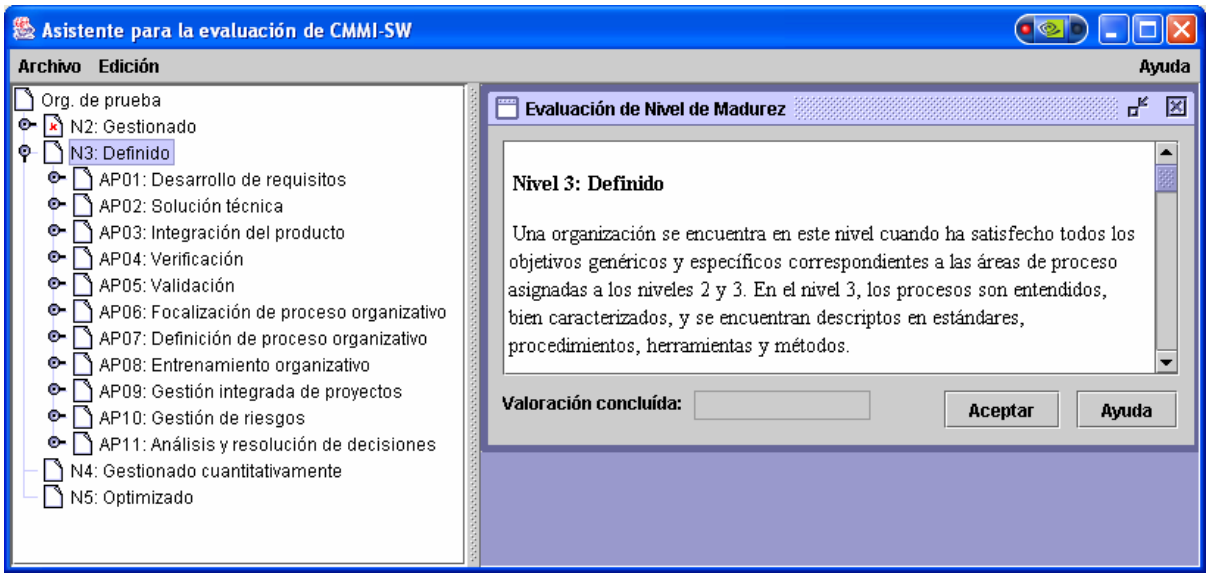

Figura 13.18. Ventana principal de la aplicación luego de iniciar la evaluación del nivel 3. En el árbol de navegación del modelo puede verse el nivel 2 marcado con una cruz roja, lo que significa que hay elementos pendientes de revisión.

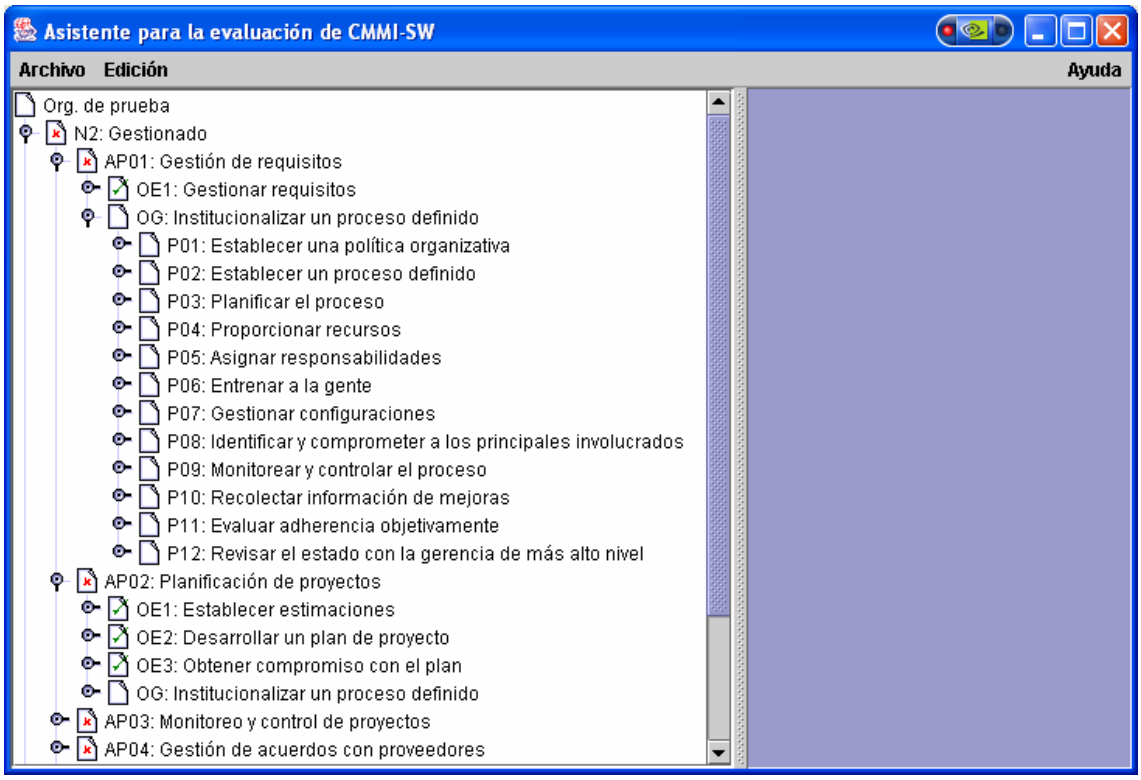

Figura 13.19. Ventana principal de la aplicación luego de iniciar la evaluación del nivel 3. En el árbol de navegación del modelo puede verse que se han modificado los objetivos genéricos del nivel 2. Los nuevos objetivos deben ser reevaluados junto con la evaluación del nivel 3.

Evaluación de Práctica

Para evaluar una práctica, se debe seleccionar la misma en el árbol de navegación del modelo, y a continuación la opción "Evaluar" del menú Edición, o bien, botón derecho del mouse y opción "Evaluar". Las prácticas pueden evaluarse a nivel de instancia o a nivel de conjunto de instancias. La figura 13.20 muestra la ventana de Evaluación de práctica para el caso de evaluación de una instancia.

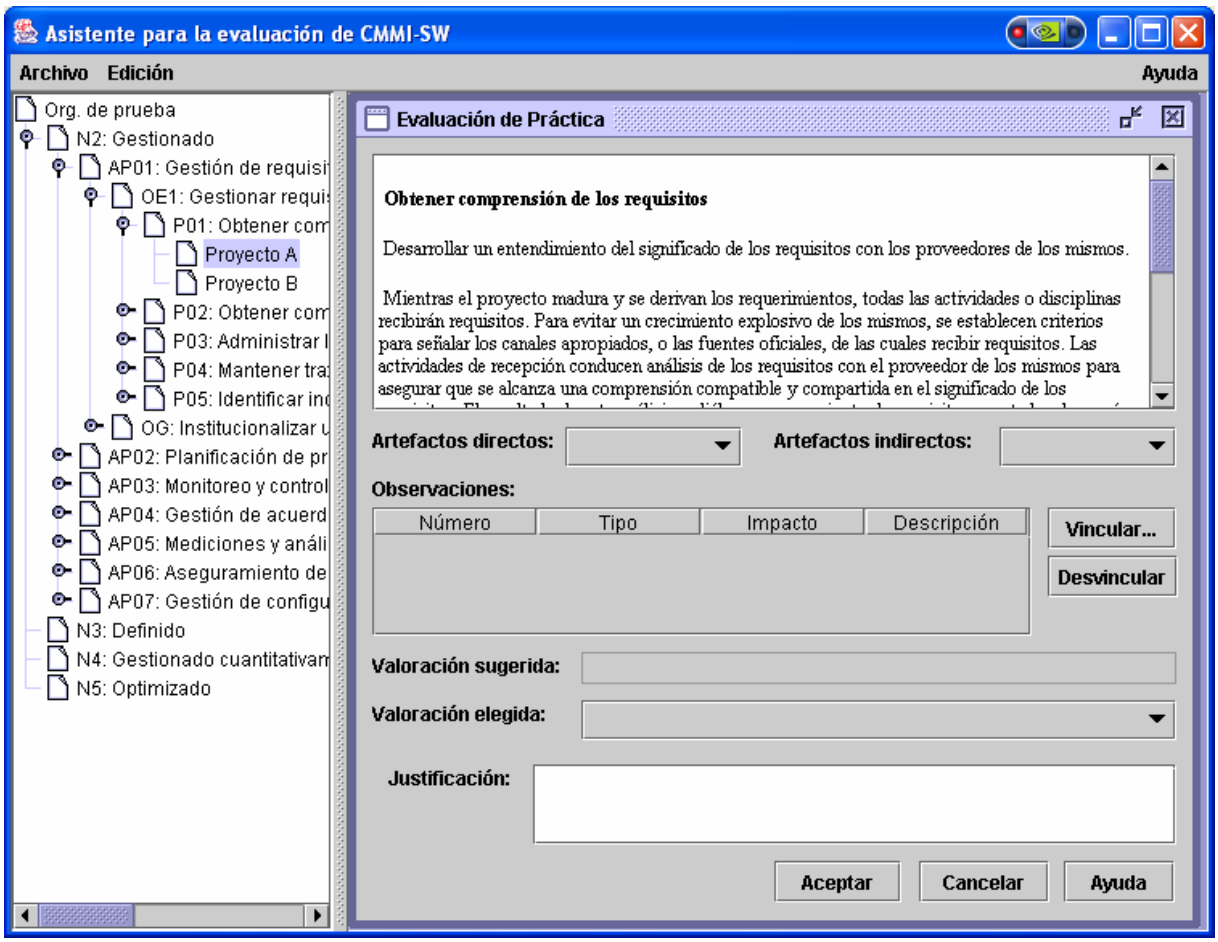

Figura 13.20. Ventana principal de la aplicación mostrando la ventana de Evaluación de práctica. En la ventana puede verse la guía online para la evaluación, y los parámetros a indicar para su evaluación.

Cuando se evalúa una práctica a nivel de instancia, la ventana cuenta con los siguientes parámetros:

- **Artefactos directos**: son los elementos tangibles que resultan directamente de la implementación de la práctica. Ejemplos de estos elementos son: documentos, entregables, materiales de entrenamiento, etc. Los valores posibles son:

"Apropiados": existen artefactos directos y son apropiados

"No apropiados": existen y no son apropiados o bien que no existen
"No se sabe": no se sabe si existen o no artefactos directos

- **Artefactos indirectos**: son elementos que surgen como consecuencia de la implementación de la práctica, pero que no constituyen en sí el propósito de la práctica. Ejemplos de estos elementos son: minutas de reunión, revisión de resultados, reportes de estado, mediciones de desempeño, etc. Los valores posibles son:

  "Confirman": existen artefactos indirectos que confirman la práctica
  "Sugieren": existen artefactos indirectos que sugieren la práctica
  "No se sabe": no se sabe si existen o no artefactos indirectos

- **Observaciones**: son elementos que se recopilan durante el análisis de una organización o proyecto. Las observaciones resaltan las fortalezas y debilidades encontradas en la organización con respecto a la implementación de las prácticas. Los botones Vincular y Desvincular permiten vincular y desvincular las observaciones existentes con la práctica bajo evaluación.

Al presionar el botón Vincular, se abre la ventana de Vinculación de observaciones. La misma se muestra en la figura 13.21.

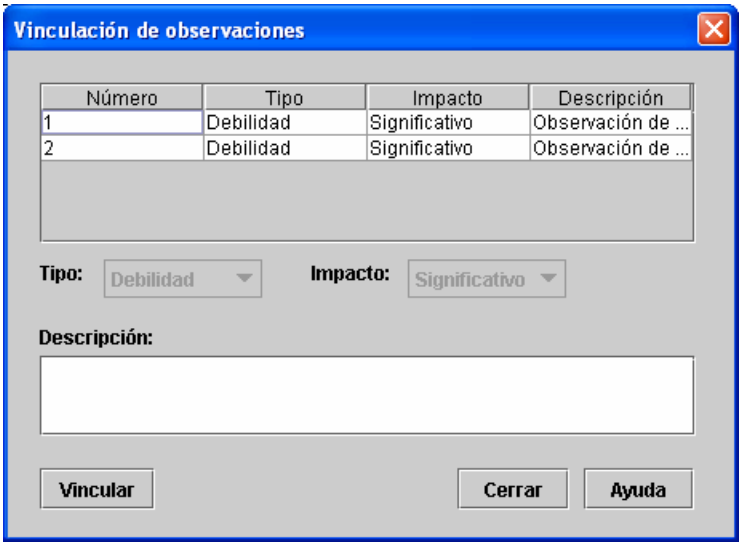

Figura 13.21. Ventana de Vinculación de observaciones. Al seleccionar una observación de la lista, se muestran los datos en la parte inferior de la ventana. Presionando el botón Vincular se vincula la observación seleccionada con la práctica bajo evaluación.

Para desvincular observaciones, se selecciona la observación en la lista del campo **Observaciones**, y se presiona el botón Desvincular.

- **Valoración sugerida**: muestra la valoración sugerida de acuerdo a la aplicación de las reglas del método SCAMPI sobre los valores ingresados por el usuario.
- **Valoración elegida**: permite asignar una valoración elegida.
- **Justificación**: permite ingresar una justificación vinculada a la valoración elegida.

Al seleccionar valores para los parámetros **Artefactos directos**, **Artefactos indirectos**, y **Observaciones**, el sistema genera automáticamente una **Valoración sugerida**. La figura 13.22 muestra un ejemplo.

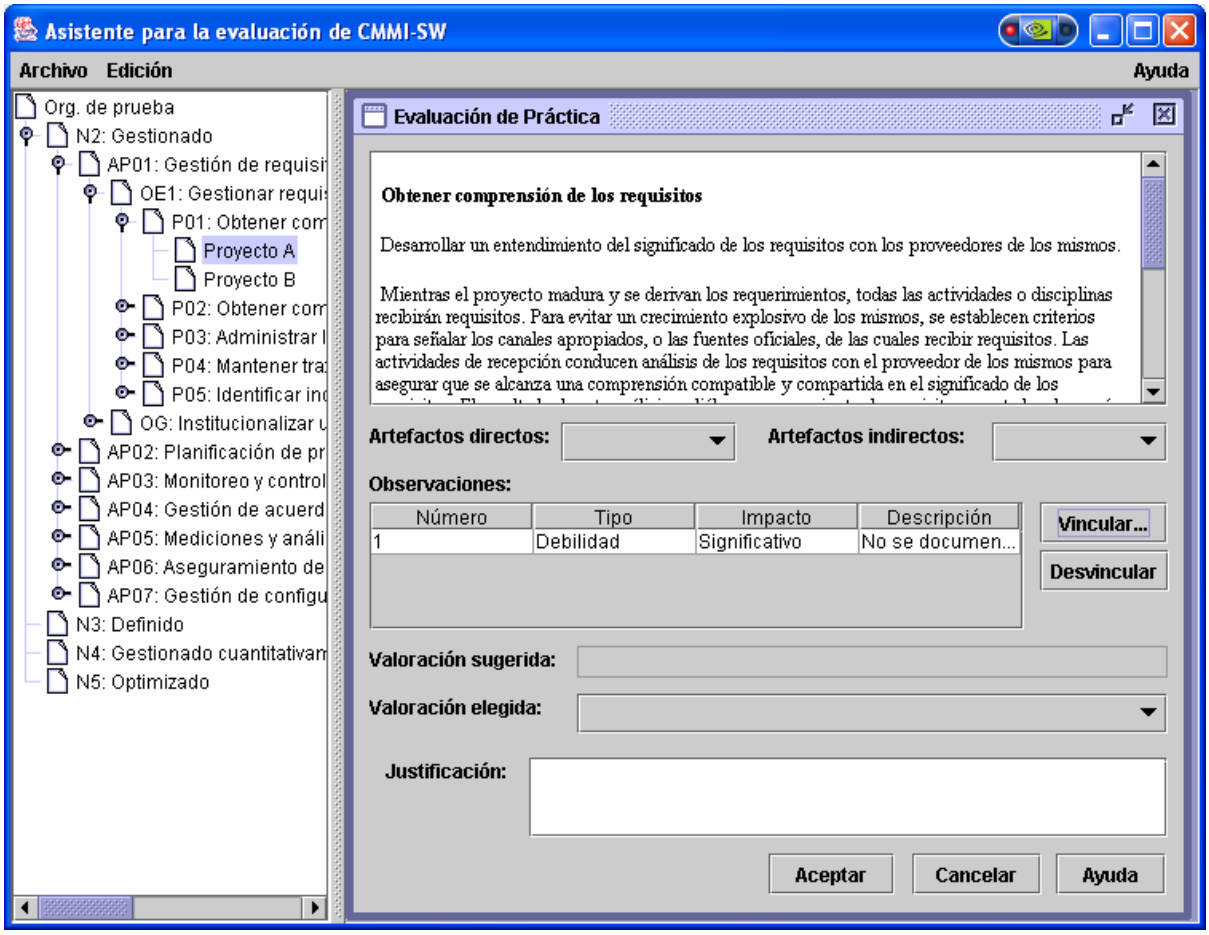

Figura 13.22. Ventana de Evaluación de práctica mostrando Valoración sugerida.

Finalmente, al indicar una **Valoración elegida** y presionar Aceptar, el sistema asigna la valoración a la práctica y cierra la ventana de evaluación. En el árbol de navegación del modelo se muestra la instancia con la marca verde de evaluada.

Cuando se terminan de evaluar todas las instancias de una práctica, el sistema genera una **Valoración sugerida** para la práctica a nivel conjunto. La figura 13.23 muestra el aspecto del árbol de navegación del modelo cuando ocurre esto.

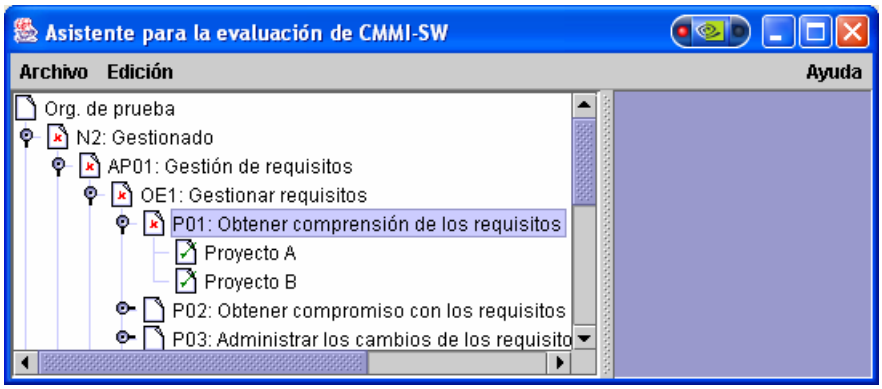

Figura 13.23. Árbol de navegación del modelo cuando se genera una Valoración sugerida para una práctica a nivel conjunto, como producto de la evaluación de sus instancias individuales.

A partir de ese momento puede evaluarse la práctica a nivel conjunto. La figura 13.24 muestra la ventana de evaluación para el caso de práctica a nivel de conjunto.

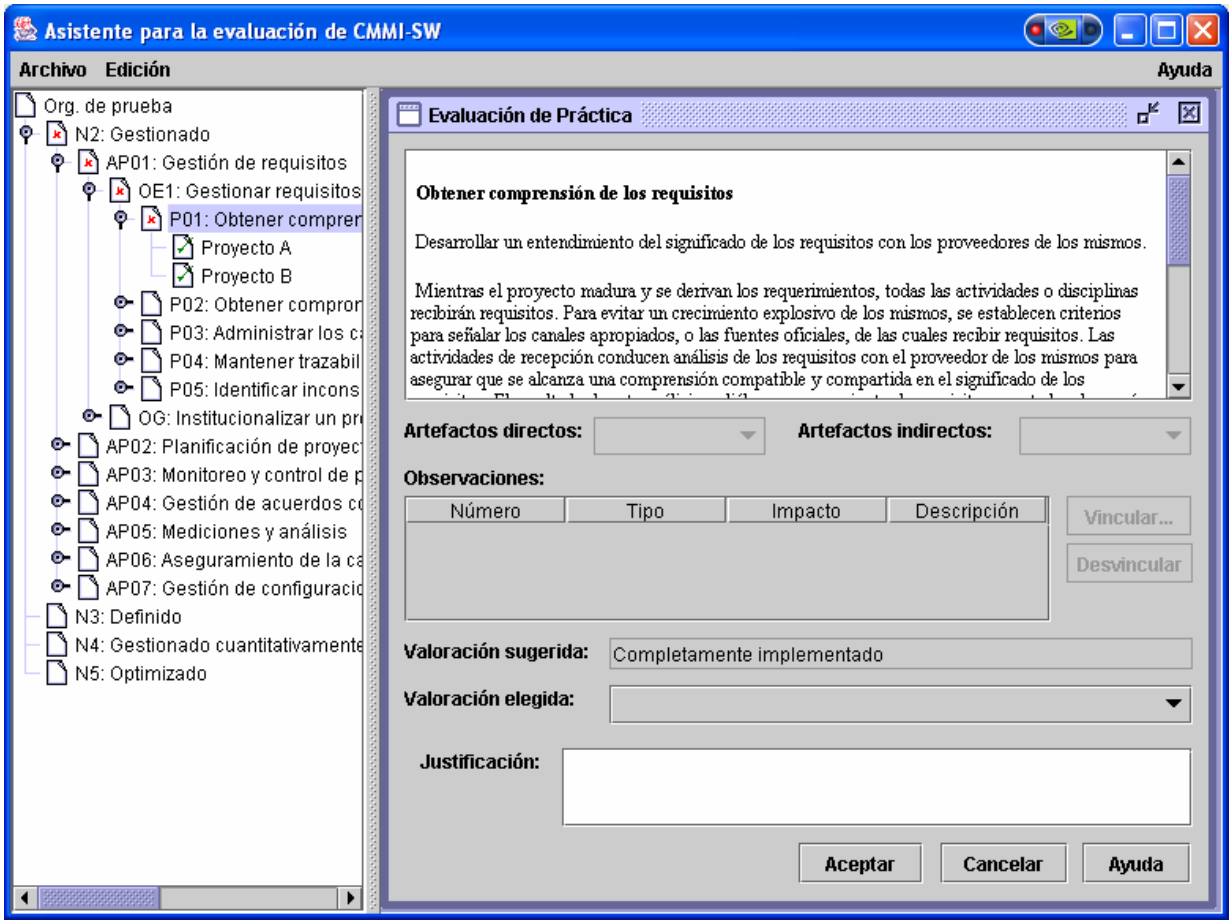

Figura 13.24. Ventana principal de la aplicación mostrando la ventana de Evaluación de práctica para una evaluación a nivel de conjunto.

Cuando se evalúa una práctica a nivel de conjunto, la ventana de Evaluación de práctica cuenta con los siguientes parámetros:

- **Valoración sugerida**: muestra la valoración sugerida de acuerdo a la aplicación de las reglas del método SCAMPI sobre las valoraciones asignadas a las instancias.

- **Valoración elegida**: permite asignar una valoración elegida.

- **Justificación**: permite ingresar una justificación vinculada a la valoración elegida.

El resto de los parámetros se encuentran inhabilitados.

Finalmente, al indicar una **Valoración elegida** y presionar Aceptar, el sistema asigna la valoración a la práctica y cierra la ventana de evaluación. En el árbol de navegación del modelo se muestra la práctica con la marca verde de evaluada.

Caso particular: Alcance a nivel de Prácticas sin instancias

Un caso particular de la evaluación de prácticas se da cuando se inicia una evaluación con alcance a nivel de prácticas pero no se definen instancias. En esta situación, la aplicación trata a la práctica como un único elemento, desapareciendo el concepto de instancia y de conjunto. El comportamiento de la ventana de Evaluación de práctica en este caso, es exactamente igual al de la evaluación de instancias.

La figura 13.25 muestra la evaluación de una práctica para la situación descripta.

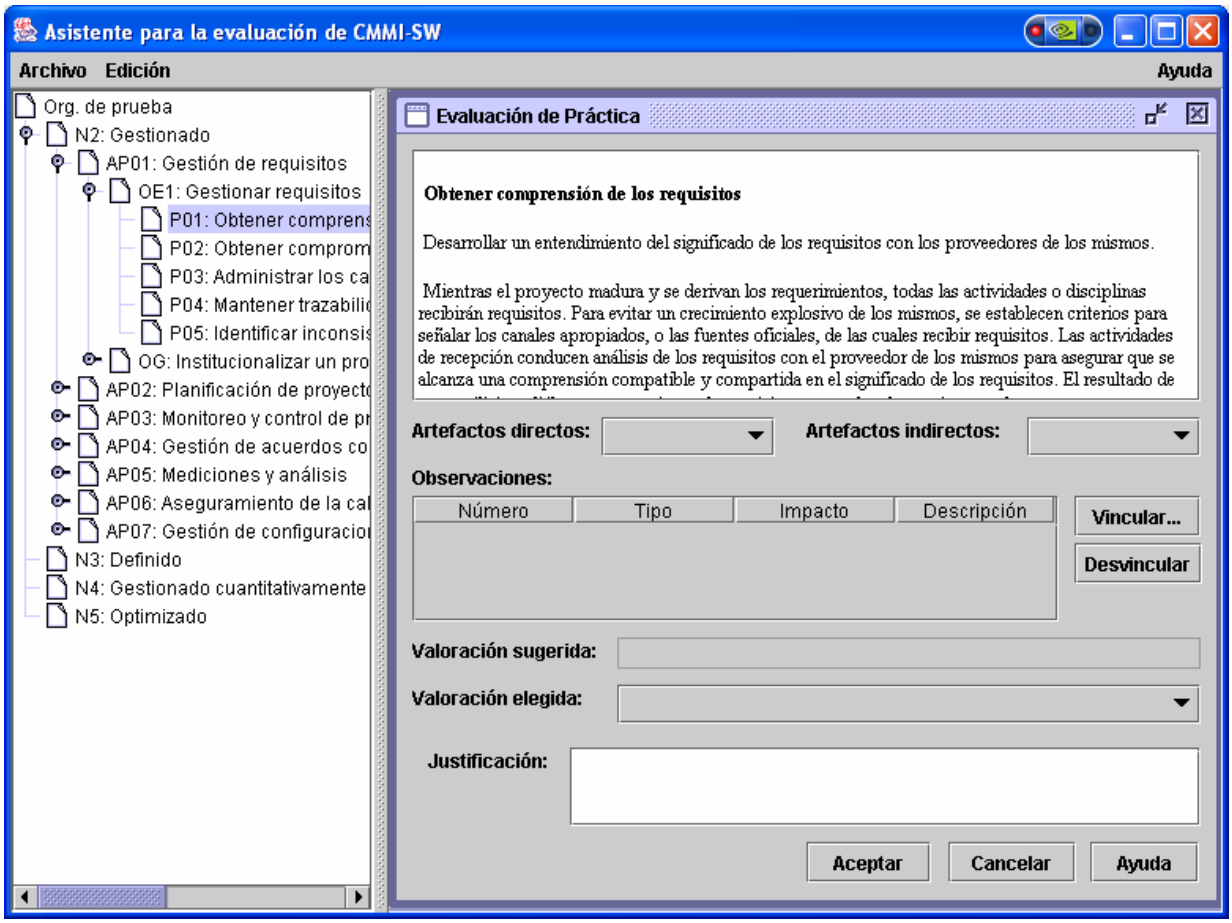

Figura 13.25. Ventana principal de la aplicación mostrando la ventana de Evaluación de práctica para el caso en el que no se definen instancias.

Evaluación de Objetivo

Para evaluar un objetivo, se debe seleccionar el mismo en el árbol de navegación del modelo, y a continuación la opción “Evaluar” del menú Edición, o bien, botón derecho del mouse y opción “Evaluar”. La figura 13.26 muestra la ventana de Evaluación de objetivo.

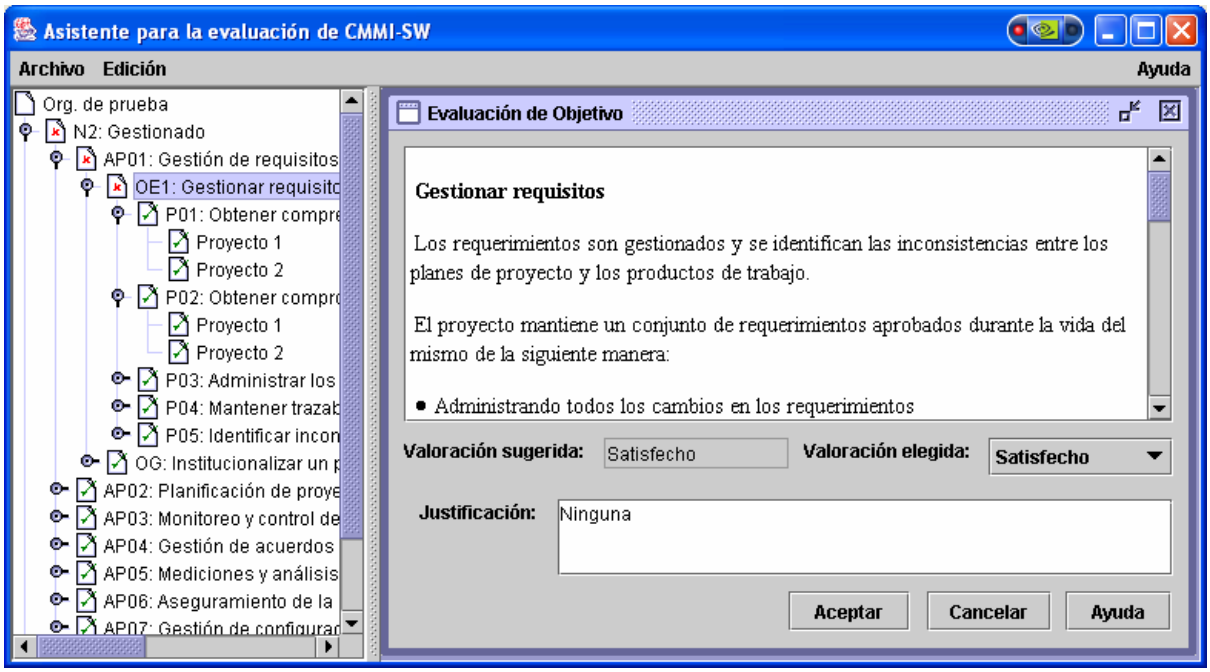

Figura 13.26. Ventana principal de la aplicación mostrando la ventana de Evaluación de objetivo con su guía online y los parámetros de evaluación. En el árbol de navegación del modelo puede verse que se han evaluado todas las prácticas asociadas al objetivo (marcas en verde), y que se ha generado una valoración sugerida para el mismo (marca en rojo). La ventana de evaluación muestra la valoración sugerida (Satisfecho).

La ventana de Evaluación de objetivo cuenta con los siguientes parámetros:

- **Valoración sugerida**: muestra la valoración sugerida de acuerdo a la aplicación de las reglas del método SCAMPI sobre las valoraciones asignadas a las prácticas.

- **Valoración elegida**: permite asignar una valoración elegida.

- **Justificación**: permite ingresar una justificación vinculada a la valoración elegida.

Al indicar una **Valoración elegida** y presionar Aceptar, el sistema asigna la valoración al objetivo y cierra la ventana de evaluación. En el árbol de navegación del modelo se muestra el objetivo con la marca verde de evaluado.

Caso particular: Alcance a nivel de Objetivos

Un caso particular de la evaluación de objetivos se da cuando se inicia una evaluación con alcance a nivel de objetivos. En esta situación, la aplicación trata al objetivo como un elemento sin hijos, con lo cual no genera valoraciones sugeridas para el mismo. El comportamiento de la ventana de Evaluación de objetivo es exactamente igual al descripto en el apartado anterior, la única diferencia radica en que el campo **Valoración sugerida** permanece vacío. La figura 13.27 muestra la evaluación de un objetivo para la situación descripta.

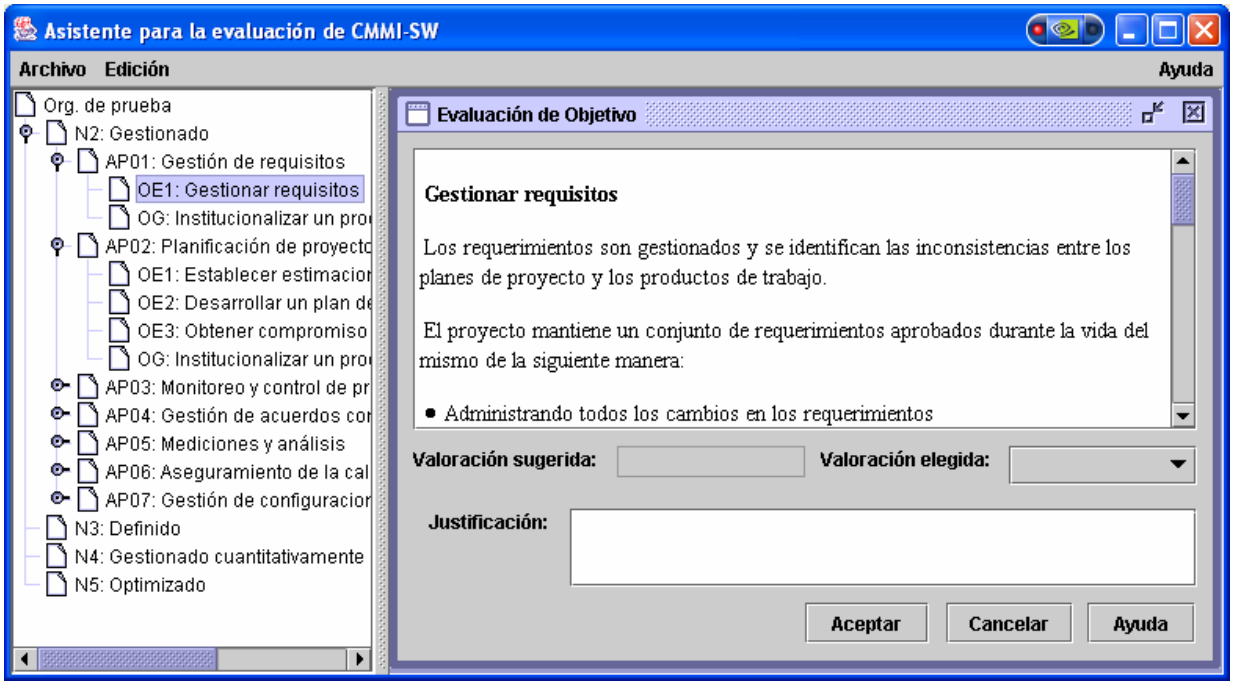

Figura 13.27. Ventana principal de la aplicación mostrando la ventana de Evaluación de objetivo para alcance a nivel de objetivos. En el árbol de navegación del modelo se puede ver que no existen prácticas por debajo de los objetivos.

Evaluación de Área de proceso

Para evaluar un área de proceso, se debe seleccionar la misma en el árbol de navegación del modelo, y a continuación la opción “Evaluar” del menú Edición, o bien, botón derecho del mouse y opción “Evaluar”. La figura 13.28 muestra la ventana de Evaluación de área de proceso.

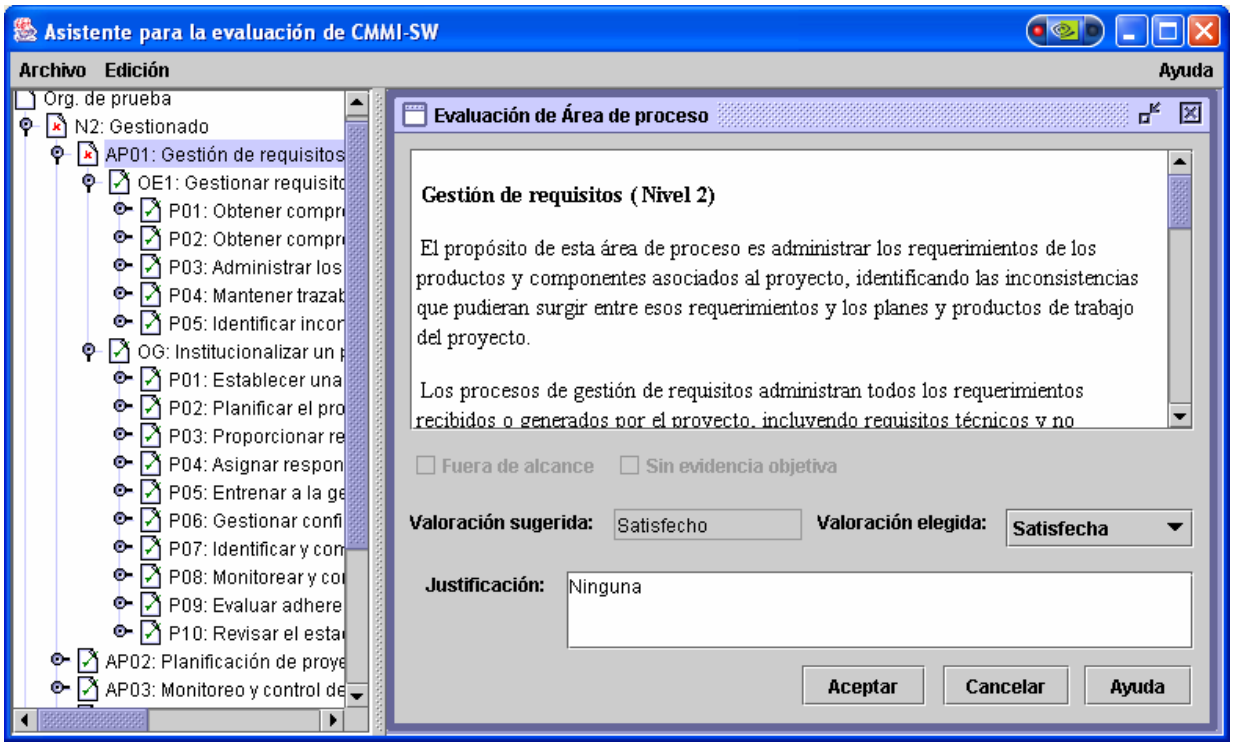

Figura 13.28. Ventana principal de la aplicación mostrando la ventana de Evaluación de área de proceso con su guía online y los parámetros de evaluación. En el árbol de navegación del modelo puede verse que se han evaluado todos los objetivos asociados al área de proceso (marcas en verde), y que se ha generado una valoración sugerida para la misma (marca en rojo). La ventana de evaluación muestra la valoración sugerida (Satisfecho).

La ventana de Evaluación de área de proceso cuenta con los siguientes parámetros:

- **Valoración sugerida**: muestra la valoración sugerida de acuerdo a la aplicación de las reglas del método SCAMPI sobre las valoraciones asignadas a los objetivos.

- **Valoración elegida**: permite asignar una valoración elegida.

- **Justificación**: permite ingresar una justificación vinculada a la valoración elegida.

Al indicar una **Valoración elegida** y presionar Aceptar, el sistema asigna la valoración al área de proceso y cierra la ventana de evaluación. En el árbol de navegación del modelo se muestra el área de proceso con la marca verde de evaluada.

Caso particular: Evaluación de Área de proceso sin evaluar Objetivos

Un caso particular de la evaluación de áreas de proceso se da cuando se desea indicar que el área de proceso no aplica a la organización, o bien cuando no hay evidencias que permitan efectuar la evaluación.

Al intentar evaluar un área de proceso sin haber evaluado sus objetivos, la aplicación muestra la advertencia mostrada en la figura 13.29.

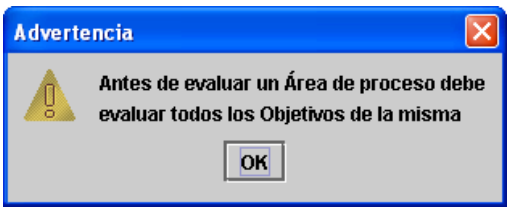

Figura 13.29. Ventana de advertencia al intentar evaluar un área de proceso sin haber evaluado sus objetivos.

Aceptando la advertencia, se abre la ventana de Evaluación de área de proceso mostrando como únicas opciones habilitadas los parámetros **Fuera de alcance** y **Sin evidencia objetiva**.

- **Fuera de alcance**: permite indicar que el área de proceso está fuera del alcance de la evaluación. En caso de indicar esta opción, el sistema genera una Valoración sugerida "No aplicable". Esta opción se debe utilizar en aquellos casos en que el área de proceso no aplique a la organización bajo análisis.

- **Sin evidencia objetiva**: permite indicar que no se ha encontrado evidencia objetiva que confirme la implementación del área de proceso en la organización bajo análisis. En caso de indicar esta opción, el sistema genera una Valoración sugerida "Sin puntaje".

Seleccionando cualquiera de ellos, se genera una Valoración sugerida y se habilitan los parámetros restantes, lo que permite asignar una Valoración elegida.

Si bien la aplicación permite asignar cualquier Valoración elegida, los valores a indicar en este caso deberían ser "No aplicable" o "Sin puntaje".

Caso particular: Alcance a nivel de Áreas de proceso

Otro caso particular de la evaluación de áreas de proceso se da cuando se inicia una evaluación con alcance a nivel de áreas de proceso. En esta situación, la aplicación trata al área de proceso como un elemento sin hijos, con lo cual no genera valoraciones sugeridas para el mismo. El comportamiento de la ventana de Evaluación de área de proceso es exactamente igual al descripto en el apartado anterior, la única diferencia radica en que el campo **Valoración sugerida** permanece vacío.

La figura 13.30 muestra la evaluación de un área de proceso para la situación descripta.

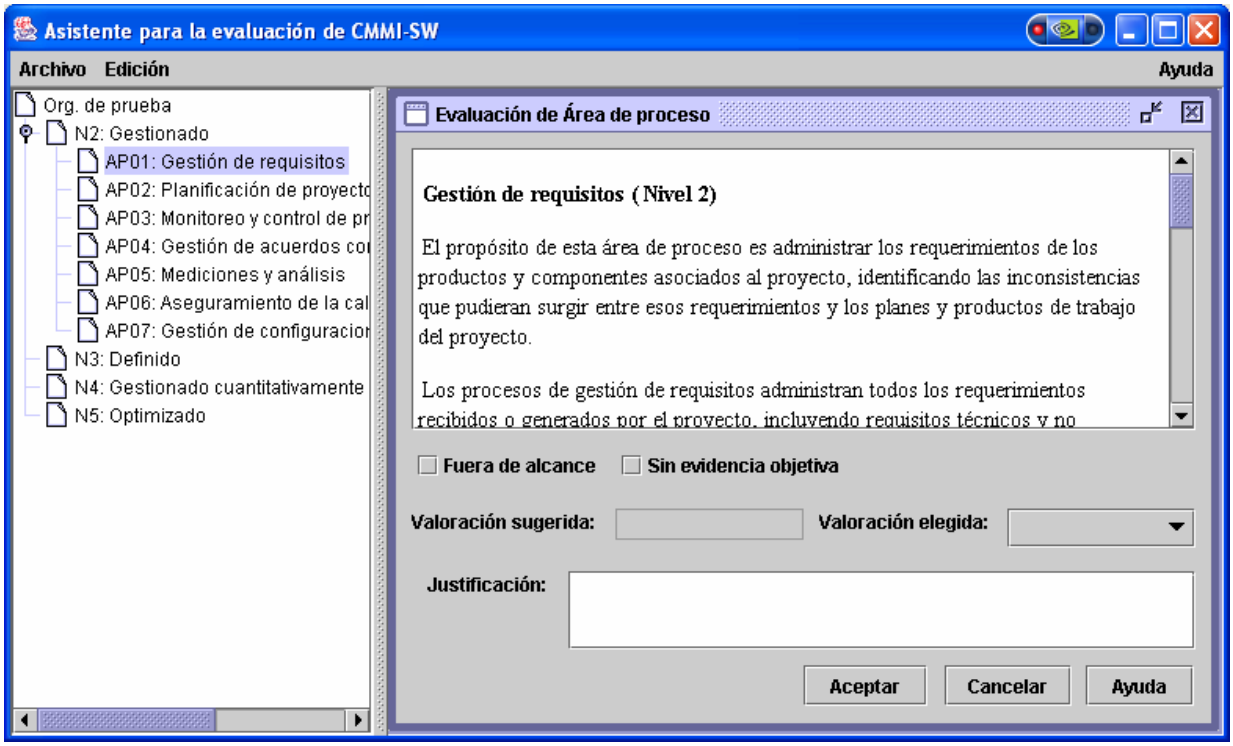

Figura 13.30. Ventana principal de la aplicación mostrando la ventana de Evaluación de área de proceso para alcance a nivel de áreas de proceso. En el árbol de navegación del modelo se puede ver que no existen objetivos por debajo de las áreas de proceso.

13.4 Bitácora del proyecto

En esta sección se vuelca un extracto de la bitácora del proyecto para cada una de las iteraciones del desarrollo. La figura 13.31 muestra el gantt real del proyecto para la iteración 1 (Viabilidad).

ID	Task Name	Duration	Start	Finish
1	**Iteración 1 - Viabilidad**	**42 days**	**Thu 6/19/03**	**Wed 7/30/03**
2	**EVS: Estudio de Viabilidad del Sistema**	**18 days**	**Thu 6/19/03**	**Sun 7/6/03**
3	EVS1 Establecimiento del Alcance del Sistema	15 days	Thu 6/19/03	Thu 7/3/03
4	EVS2 Estudio de la situación actual	3 days	Fri 7/4/03	Sun 7/6/03
5	EVS3 Definición de los requisitos del sistema	3 days	Fri 7/4/03	Sun 7/6/03
6	**Gestión del proyecto**	**17 days**	**Mon 7/7/03**	**Wed 7/23/03**
7	GPI1 Estimación de esfuerzo	6 days	Mon 7/7/03	Sat 7/12/03
8	GPI2 Planificación	11 days	Sun 7/13/03	Wed 7/23/03
9	**Gestión de la configuración**	**5 days**	**Thu 7/24/03**	**Mon 7/28/03**
10	EVS-GC1 Definición de los requisitos de GC	1 day	Thu 7/24/03	Thu 7/24/03
11	EVS-GC2 Establecimiento del plan de GC	4 days	Fri 7/25/03	Mon 7/28/03
12	**Aseguramiento de la Calidad**	**1 day**	**Tue 7/29/03**	**Tue 7/29/03**
13	EVS-CAL1 Identificación de las propiedades de calidad	1 day	Tue 7/29/03	Tue 7/29/03
14	EVS-CAL2 Establecimiento del plan de QA	1 day	Tue 7/29/03	Tue 7/29/03
15	EVS-CAL3 Adecuación del plan de QA a la solución	1 day	Tue 7/29/03	Tue 7/29/03
16	**Seguridad**	**30 days**	**Tue 7/1/03**	**Wed 7/30/03**
17	EVS-SEG1 Estudio de la seguridad requerida en EVS	1 day	Tue 7/1/03	Tue 7/1/03
18	EVS-SEG5 Evaluación de la seguridad de la solución	1 day	Wed 7/30/03	Wed 7/30/03
19	Entrega de productos Iteración 1	1 day	Fri 8/1/03	Fri 8/1/03

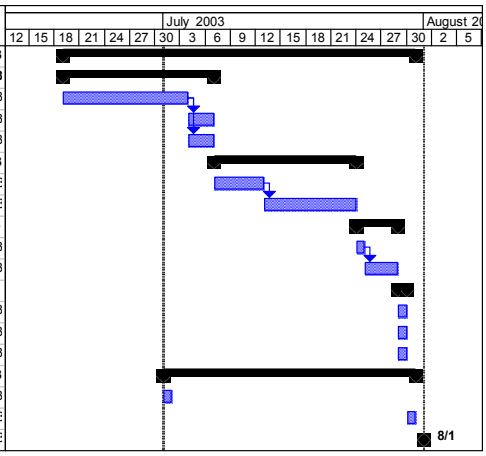

Figura 13.31. Gantt de seguimiento para la Iteración 1 (Viabilidad)

La figura 13.32 muestra el gantt real del proyecto para la iteración 2 (Arquitectura).

ID	Task Name	Duration	Start	Finish
20	**Iteración 2 - Arquitectura**	**155 days**	**Mon 8/4/03**	**Mon 1/5/04**
21	**ASI: Análisis del Sistema de Información**	**44 days**	**Mon 8/4/03**	**Tue 9/16/03**
22	ASI1 Definición del sistema	7 days	Mon 8/4/03	Sun 8/10/03
23	ASI2 Establecimiento de requisitos	6 days	Thu 8/14/03	Tue 8/19/03
24	ASI8 Definición de interfaces de usuario	3 days	Wed 8/20/03	Fri 8/22/03
25	ASI4 Análisis de los casos de uso	6 days	Sat 8/23/03	Thu 8/28/03
26	ASI5 Análisis de las clases	6 days	Sat 8/23/03	Thu 8/28/03
27	ASI9 Análisis de consistencia y especif. de requisitos	5 days	Fri 9/5/03	Tue 9/9/03
28	ASI10 Especificación del plan de pruebas	5 days	Fri 9/12/03	Tue 9/16/03
29	**DSI: Diseño del Sistema de Información**	**102 days**	**Sat 9/20/03**	**Tue 12/30/03**
30	DSI1 Definición de la arquitectura del sistema	5 days	Sat 9/20/03	Wed 9/24/03
31	DSI2 Diseño de la arquitectura de soporte	5 days	Thu 9/25/03	Mon 9/29/03
32	DSI3 Diseño de casos de uso reales	19 days	Sat 10/4/03	Wed 10/22/03
33	DSI4 Diseño de clases	19 days	Sat 10/4/03	Wed 10/22/03
34	DSI7 Verificación y aceptación de la arq. del sistema	3 days	Thu 10/23/03	Sat 10/25/03
35	DSI8 Generación de especificaciones de construcción	1 day	Mon 10/27/03	Mon 10/27/03
36	DSI10 Especificación técnica del plan de pruebas	26 days	Tue 10/28/03	Tue 12/30/03
37	**CSI: Construcción del Sistema de Información**	**70 days**	**Tue 10/28/03**	**Mon 1/5/04**
38	CSI1 Preparación del entorno de generación y construcci	1 day	Tue 10/28/03	Tue 10/28/03
39	CSI2 Generación del código de los componentes y proce	43 days	Thu 10/30/03	Thu 12/11/03
40	CSI3 Ejecución de las pruebas unitarias	43 days	Thu 10/30/03	Thu 12/11/03
41	CSI4 Ejecución de las pruebas de integracion	43 days	Thu 10/30/03	Thu 12/11/03
42	CSI5 Ejecución de las pruebas del sistema	2 days	Wed 12/31/03	Thu 1/1/04
43	CSI6 Elaboración de los manuales de usuario	4 days	Fri 1/2/04	Mon 1/5/04
44	**Gestión de la configuración**	**155 days**	**Mon 8/4/03**	**Mon 1/5/04**
45	GC1 Identificación y registro de productos	155 days	Mon 8/4/03	Mon 1/5/04
46	**Aseguramiento de la calidad**	**118 days**	**Wed 9/10/03**	**Mon 1/5/04**
47	ASI-CAL2 Especif. detallada del plan de aseg. de calidad	2 days	Wed 9/10/03	Thu 9/11/03
48	ASI-CAL3 Revisión del análisis de consistencia	2 days	Thu 9/11/03	Fri 9/12/03
49	ASI-CAL4 Revisión del plan de pruebas	1 day	Tue 9/16/03	Tue 9/16/03
50	DSI-CAL1 Revisión de la verificación de la arquitectura d	1 day	Sun 10/26/03	Sun 10/26/03
51	DSI-CAL2 Revisión de la especificación técnica del plan	1 day	Tue 12/30/03	Tue 12/30/03
52	CSI-CAL2 Revisión de las pruebas unitarias, de integraci	1 day	Fri 1/2/04	Fri 1/2/04
53	CSI-CAL3 Revisión de los manuales de usuario	1 day	Mon 1/5/04	Mon 1/5/04
54	Entrega de productos Iteración 2	1 day	Tue 1/6/04	Tue 1/6/04

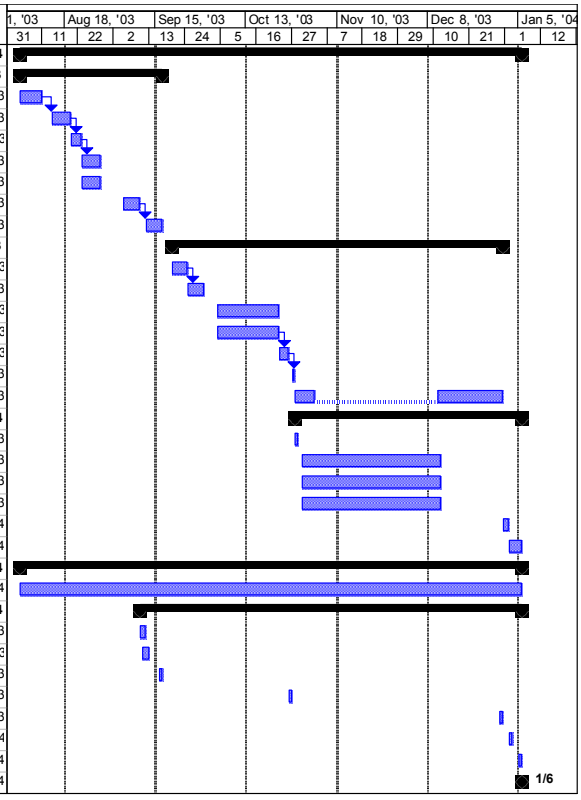

Figura 13.32. Gantt de seguimiento para la Iteración 2 (Arquitectura)

La figura 13.33 muestra el gantt real del proyecto para la iteración 3 (Construcción).

ID	Task Name	Duration	Start	Finish
55	**Iteración 3 - Construcción**	**20 days**	**Wed 1/7/04**	**Mon 1/26/04**
56	**ASI: Análisis del Sistema de Información**	**2 days**	**Wed 1/7/04**	**Thu 1/8/04**
57	ASI2 Establecimiento de requisitos	1 day	Wed 1/7/04	Wed 1/7/04
58	ASI8 Definición de interfaces de usuario	1 day	Wed 1/7/04	Wed 1/7/04
59	ASI4 Análisis de los casos de uso	1 day	Thu 1/8/04	Thu 1/8/04
60	ASI5 Análisis de las clases	1 day	Thu 1/8/04	Thu 1/8/04
61	ASI9 Análisis de consistencia y especif. de requisitos	1 day	Thu 1/8/04	Thu 1/8/04
62	**DSI: Diseño del Sistema de Información**	**6 days**	**Fri 1/9/04**	**Wed 1/14/04**
63	DSI1 Definición de la arquitectura del sistema	1 day	Fri 1/9/04	Fri 1/9/04
64	DSI2 Diseño de la arquitectura de soporte	2 days	Sat 1/10/04	Sun 1/11/04
65	DSI3 Diseño de casos de uso reales	1 day	Mon 1/12/04	Mon 1/12/04
66	DSI4 Diseño de clases	4 days	Fri 1/9/04	Mon 1/12/04
67	DSI7 Verificación y aceptación de la arq. del sistema	2 days	Tue 1/13/04	Wed 1/14/04
68	DSI8 Generación de especificaciones de construcción	1 day	Wed 1/14/04	Wed 1/14/04
69	DSI10 Especificación técnica del plan de pruebas	1 day	Wed 1/14/04	Wed 1/14/04
70	**CSI: Construcción del Sistema de Información**	**12 days**	**Thu 1/15/04**	**Mon 1/26/04**
71	CSI2 Generación del código de los componentes y proce	11 days	Thu 1/15/04	Sun 1/25/04
72	CSI3 Ejecución de las pruebas unitarias	11 days	Thu 1/15/04	Sun 1/25/04
73	CSI4 Ejecución de las pruebas de integracion	11 days	Thu 1/15/04	Sun 1/25/04
74	CSI5 Ejecución de las pruebas del sistema	2 days	Sun 1/25/04	Mon 1/26/04
75	CSI6 Elaboración de los manuales de usuario	1 day	Mon 1/26/04	Mon 1/26/04
76	**Gestión de la configuración**	**20 days**	**Wed 1/7/04**	**Mon 1/26/04**
77	GC1 Identificación y registro de productos	20 days	Wed 1/7/04	Mon 1/26/04
78	**Aseguramiento de la calidad**	**19 days**	**Thu 1/8/04**	**Mon 1/26/04**
79	ASI-CAL3 Revisión del análisis de consistencia	1 day	Thu 1/8/04	Thu 1/8/04
80	DSI-CAL1 Revisión de la verificación de la arquitectura d	1 day	Wed 1/14/04	Wed 1/14/04
81	DSI-CAL2 Revisión de la especificación técnica del plan	1 day	Wed 1/14/04	Wed 1/14/04
82	CSI-CAL2 Revisión de las pruebas unitarias, de integraci	1 day	Mon 1/26/04	Mon 1/26/04
83	CSI-CAL3 Revisión de los manuales de usuario	1 day	Mon 1/26/04	Mon 1/26/04
84	Entrega de productos Iteración 3	1 day	Mon 1/26/04	Mon 1/26/04

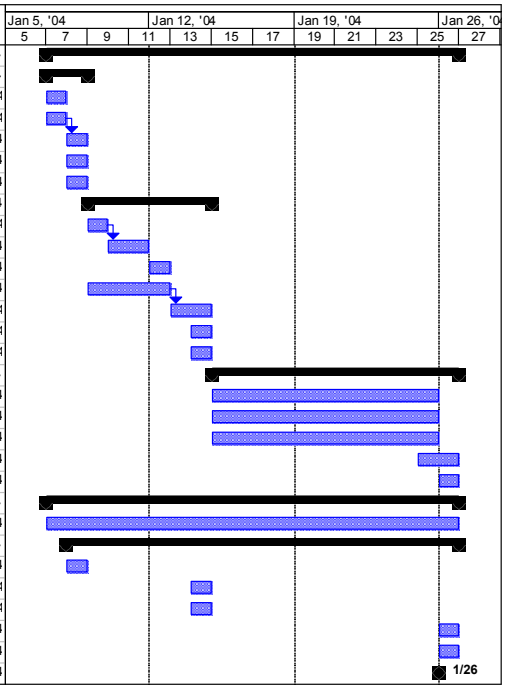

Figura 13.33. Gantt de seguimiento para la Iteración 3 (Construcción)

La figura 13.34 muestra el gantt real del proyecto para la iteración 4 (Cierre).

ID	Task Name	Duration	Start	Finish
85	**Iteración 4 - Cierre**	**12 days**	**Tue 1/27/04**	**Sat 2/7/04**
86	**IAS: Implantación y Aceptación del Sistema**	**2 days**	**Tue 1/27/04**	**Wed 1/28/04**
87	IAS3 Incorporación del sistema al entorno de operación	1 day	Tue 1/27/04	Tue 1/27/04
88	IAS6 Pruebas de aceptación del sistema	2 days	Tue 1/27/04	Wed 1/28/04
89	**Gestión de la configuración**	**2 days**	**Tue 1/27/04**	**Wed 1/28/04**
90	GC1 Identificación y registro de productos	2 days	Tue 1/27/04	Wed 1/28/04
91	**Aseguramiento de la calidad**	**1 day**	**Wed 1/28/04**	**Wed 1/28/04**
92	IAS-CAL3 Revisión de las pruebas de aceptación del sist	1 day	Wed 1/28/04	Wed 1/28/04
93	Elaboración del documento de tesis	11 days	Wed 1/28/04	Sat 2/7/04
94	Entrega de productos Iteración 4	1 day	Mon 2/9/04	Mon 2/9/04

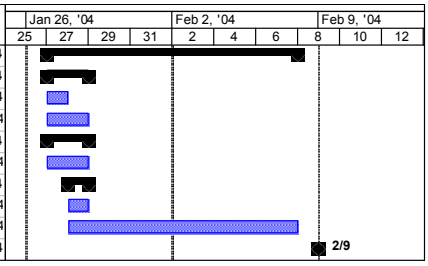

Figura 13.34. Gantt de seguimiento para la Iteración 4 (Cierre)