ArgoUML: Manual de Usuario

Tutorial y descripción de referencia

Alejandro Ramirez
Philippe Vanpeperstraete
Andreas Rueckert
Kunle Odutola
Jeremy Bennett
Linus Tolke
Michiel van der Wulp

ArgoUML: Manual de Usuario : Tutorial y descripción de referencia

por Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola, Jeremy Bennett, Linus Tolke, y Michiel van der Wulp

Resumen

Esta version de el manual está planeado para describir la versión 0.34 de ArgoUML.

Tabla de contenidos

Prefacio

El diseño de software es una tarea cognitiva dificil. Los diseñadores deben construir manualmente diseños, pero la dificultad principal es la toma de decisiones en lugar de la entrada de datos. Si los diseñadores mejoran sus capacidades de toma de decisiones, ello resultaría en mejores diseños.

Las herramientas CASE actuales proporcionan automatización e interfaces gráficos de usuario que reducen el trabajo manual de construir un diseño y transformar un diseño en codigo. Ayudan a los diseñadores en la toma de decisiones principalmente proporcionando visualización de los diagramas de diseño y comprobaciones sintacticas simples.. Adenas muchas herramientas CASE proporcionan beneficios sustanciales en el area de control de versiones y mecanismos de diseño concurrente. Un area de soporte de diseño que no ha sido bien soportada es el analisis de decisiones de diseño.

Las herramientas CASE actuales son utiles en que proporcionan una GUI (Graphic User Interface; Interfaz Grafica de Usuario) que permite a los diseñadores acceder a todas las caracteristicas proporcionadas por la herramienta. Y soportan el proceso de diseño en que permiten al diseñador construir diagramas en el estilo de metodologias de diseño populares. Pero tipicamente no porporcionan soporte de proceso para guiar al diseñador a traves de la tarea de diseño. En su lugar, los diseñadores tipicamente comienzan con una página en blanco y deben recordar cubrir todos los aspectos del diseño.

ArgoUML es un entorno de diseño orientado a dominio que proporciona soporte cognitivo de diseño orientado a objetos. ArgoUML proporciona algunas de las mismas caracteristicas de automatización de una herramienta CASE comercial, pero esta enfocado en caracteristicas que soportan las necesidades cognitivas de los diseñadores. Estas necesidades cognitivas estan descritas por tres teorias cognitivas.

1. reflection-in-action;
2. opportunistic design; and
3. comprehension and problem solving.

ArgoUML esta basado en la especificación UML 1.4 . El nucleo del modelo de repositorio es una implementación de el Java Metadata Interface (JMI) que directamente soporta MOF y usa la versión legible por maquina de la especificación UML 1.4 proporcionada por OMG.

Ademas, es nuestra meta proporcionar soporte exhaustivo para OCL (el Object Constraint Language) y XMI (el formato XML Model Interchange).

ArgoUML fue originariamente desarrollado por un pequeño grupo de gente como un proyecto de investigación. ArgoUML tiene muchas caracteristicas que lo hacen especial, pero no implementa todas las caracteristicas que una herramienta CASE comercial proporciona.

La publicación V0.20 actual de ArgoUML, implementa todos los tipos de diagramas de UML 1.4 standard [http://www.omg.org/cgi-bin/doc?formal/01-09-67] (versiones de ArgoUML anteriores a 0.20 implementaban la UML 1.3 standard [http://www.omg.org/cgi-bin/doc?formal/00-03-01]). Está escrito en Java y funciona en todo sistema que proporcione una plataforma Java 2 de Java 1.4 o posterior. Usa formatos de archivo abiertos XMI [http://www.omg.org/cgi-bin/doc?formal/02-01-01] (formato XML Metadata Interchange) (para la información de modelos) y PGML [http://www.w3.org/TR/1998/NOTE-PGML] (Precision Graphics Markup Language) (para información gráfica) para almacenamiento. Cuando ArgoUML implemente UML 2.0, PGML sera sustituido por la especificación UML Diagram Interchange.

Este manual es el trabajo acumulativo de muchas personas y ha estado evolucionando durante muchos años. Conectado con la publicación 0.10 de ArgoUML, Jeremy Bennett, escribió gran cantidad de nuevo material que fue añadido a las vestiones anteriores por Alejandro Ramirez, Philippe Vanpeperstraete y Andreas Rueckert. El ademas añadió cosas de algunos de los otros documentos como el libro de cocina de los desarrolladores por Markus Klink y Linus Tolke, la Guia Rápida por Kunle Odutola, y el FAQ (Preguntas frecuentes) por Dennis Daniels. Conectado con la publicación 0.14 se realizaron cambios por Linus Tolke, y Michiel van der Wulp. Estos cambios fueron mayoritariamente adaptar el manual a las nuevas funciones y apariencia de la version 0.14 de ArgoUML, y la introducción del indice. Los usuarios y desarrolladores que han contribuido proporcionando ayuda valiosa, como revisiones, comentarios y observaciones mientras leen y usan este manual son demasiados para ser nombrados.

ArgoUML esta disponible gratuitamente y puede ser usado en entornos comerciales. Para los terminos de uso, mira el acuerdo de licencia presentado cuando tu descargas ArgoUML. Estamos proporcionando el codigo fuente de ArgoUML para que puedas revisarlo, adecuarlo a tus necesidades y mejorarlo. Pasado el tiempo, esperamos que ArgoUML evolucione en una poderosa y util herramienta que todos puedan usar.

Este Manual de Usuario esta orientado al diseñador, quien desea hacer uso de ArgoUML. El manual esta escrito asumiendo familiaridad con UML, pero eventualmente puede ayudar a aquellos nuevos en UML.

El manual esta escrito en DocBook/XML y esta disponible como HTML y PDF.

El proyecto ArgoUML da la bienvenida a aquellos que quieren estar mas involucrados. Mira en project website [http://argouml.tigris.org/] para encontrar mas información.

¡Dinos que piensas sobre este Manual de Usuario! Tus comentarios nos ayudaran a mejorar cosas. Mira Sección 1.3.3, “ Retroalimentación por el Usuario ”.

Capítulo 1. Introducción

1.1. Origenes y Visión General de ArgoUML

1.1.1. Analisis Orientado a Objeto y Diseño

Durante la última decada, el Analisis Orientado a Objeto y Diseño (Object Oriented Analysis and Design; OOA&D) se ha convertido en *el* paradigma de desarrollo de software dominante. Con ello se ha conseguido un gran avance en los procesos de pensamiento de todos los involucrados en el ciclo de vida del desarrollo des software.

El soporte de objetos en un lenguaje de programación empezó con Simula 67, pero fue la aparición en la decada de 1980 de los lenguajes híbridos, como es C++, Ada y Object Pascal lo que permitió a OOA&D despegar. Estos lenguajes proporcionaban soporte para OO ademas de para programación procedural. La *programación* Orientada a Objeto se convirtió en la corriente dominante.

Un sistema OO está diseñado y implementado como una *simulación* del mundo real usando artefactos software. Esta premisa es tan potente como simple. Usando un acercamiento OO para *diseñar* un sistema puede ser diseñado y testeado (o mas correctamente simulado) sin tener que construir el sistema real primero.

Es el desarrollo durante la decada de 1990 de herramientas para soportar *analisis* Orientado a Objeto y *diseño* lo que coloco este enfoque en la corriente dominante. Cuando se combina con la capacidad de diseñar sistemas a muy alto nivel, una herramienta basada en el enfoque OOA&D ha permitido la implementación de sistemas mas complejos que los posibles previamente.

El último factor que ha propulsado OOA&D ha sido su idoneidad para modelar interfaces graficos de usuario. La popularidad de lenguajes graficos orientados a objeto y basados en objeto como Visual Basic y Java refleja la efectividad de este enfoque.

1.1.2. El Desarrollo de ArgoUML

Durante la decada de 1980 un número de metodologías de procesos OOA&D y notaciones fueron desarrolladas por diferentes equipos de investigación. Se hizo patente que habia muchos temas comunes y, durante la decada de 1990, un enfoque unificado para la notación OOA&D fué desarrollado bajo el auspicio del Object Management Group [http://www.omg.org]. Este estandar se hizo conocido como el Unified Modeling Language (UML), y ahora es el lenguaje estandar para comunicar conceptos OO.

ArgoUML fué concebido como un entorno y herramienta para usar en el analisis y diseño de sistemas de software orientados a objeto. En este sentido es similar a muchos de las herramientas CASE comerciales que son vendidas como herramientas para modelar sistemas software. ArgoUML tiene un número de distinciones muy importantes de muchas de esas herramientas.

1. Es gratis.

2. ArgoUML se enfoca en investigación en psicología cognitiva para proporcionar nuevas caracteristicas que incrementen la productividad soportando las necesidades cognitivas de diseñadores y arquitectos de software orientado a objeto.

3. ArgoUML soporta estandares abiertos extensivamente—UML, XMI, SVG, OCL y otros.

4. ArgoUML es una aplicación Java pura 100%. Esto permite a ArgoUML funcionar en todas las plataformas para las cuales un puerto fiable de la plataforma Java 2 esta disponible.

5. ArgoUML es un proyecto de codigo abierto. La disponibilidad del codigo fuente asegura que una nueva generación de diseñadores de software e investigadores ahora tienen un entorno de trabajo probado desde el que pueden conducir el desarrollo y evolución de tecnologías de herramientas CASE.

UML es el lengueje de modelado OO mas prevalente y java es una de las plataformas de desarrollo OO mas productivas. Jason Robbins y el resto de su equipo de investigación en la universidad de California, Irvine potenciaron estos beneficios creando ArgoUML. El resultado es un entorno y una herramienta de desarrollo sólida para diseño de sistemas OO. Es mas, proporciona un campo de de pruebas para la evolución del desarrollo e investigación de herramientas CASE orientadas a objeto.

Una primera publicación de ArgoUML fue disponible en 1998 y mas de 100,000 descargas a mediados de 2001 demostró el impacto que este proyecto ha provocado, siendo popular en campos educacionales y comerciales.

1.1.3. Encontrando Mas Sobre el Proyecto ArgoUML

1.1.3.1. Como está desarrollado ArgoUML

Jason Elliot Robbins fundó el Proyecto Argo y proporciono un liderazgo temprano al proyecto. Mientras Jason permanece activo en el proyecto, él ha dejado el liderazgo. El proyecto continua avanzando fuertemente. Hay mas de 300 miembros en la lista de correo de desarrollador (mira http://argouml.tigris.org/servlets/ProjectMailingListList [http://argouml.tigris.org/servlets/ProjectMailingListList]), Con un par de docenas de ellos formando el nucleo del grupo de desarrollo.

La lista de correo del desarrollador es el lugar donde toda la discusión sobre las ultimas tareas toma lugar, y los desarrolladores discuten las direcciones que el proyecto debería tomar. Aunque controvertido a veces, estas discusiones son mantenidas siempre correctas y amigables (sin flame-wars y esas cosas), así que los novatos (newbies) no deberian dudar y participar en ellas. Siempre tendras una calurosa bienvenida allí.

Si quires aprender como se gestiona el proyecto y como contribuir a él, vete a ArgoUML Web Site Developer Zone [http://argouml.tigris.org/dev.html] y busca a traves de la documentación allí expuesta.. El Libro de Cocina del Desarrollador (Developers' Cookbook) fué escrito especificamente para este proposito.

1.1.3.2. Mas sobre la Infraestructura

Ademas de la lista de correo del desarrollador, existe tambien una lista de correo para usuarios (mira The ArgoUML Mailing List List [http://argouml.tigris.org/servlets/ProjectMailingListList]), donde podemos discutir problemas desde la perspectiva del usuario. Los desarrolladores tambien leen esa lista, así que generalmente se proporciona ayuda altamente cualificada.

Antes de postear en esta lista, deberias echar un vistazo al user FAQ [http://argouml.tigris.org/faqs/users.html] mantenido por Ewan R. Grantham.

Mas información sobre ArgoUML y otros asuntos relacionados con UML está tambien disponible en el ArgoUML website [http://argouml.tigris.org], mantenido por Linus Tolke.

1.2. Alcance de Este Manual de Usuario

1.2.1. Audiencia Objetivo

La publicación altual de este documento esta dirigida a usuarios experimentados de UML en OOA&D (quizas con otras herramientas) que desean cambiar a ArgoUML.

Publicaciones futuras soportaran diseñadores que conocen OOA&D, y desean adoptar la notación UML dentro de su proceso de desarrollo.

Un objetivo a largo plazo es soportar i) aquellos que están aprendiendo diseño y desean empezar con un proceso OOA&D que usa notación UML, y ii) gente interesada en diseño de codigo modularizado con un GUI.

1.2.2. Alcance

La intención es que este documento proporcionará una guia exhaustiva, permitiendo a los diseñadores usar ArgoUML en toda su extensión. Esto es en dos partes.

- Un manual tutorial, mostrando como trabajar con ArgoUML
- Un manual de referencia completo, registrando todo lo que puedes hacer con ArgoUML.

La version 0.22 de este documento lleva a cabo la segunda de ellas.

En esta guia hay algunas cosas que no encontraras, porque están cubiertas en otro lugar.

- Descripciones de como ArgoUML funciona internamente.
- Como mejorar ArgoUML con nuevas caracteristicas y funciones.
- Una guia de solución de problemas.
- Un indice de referencia rápida para usar ArgoUML.

Estos están cubiertos en the Developers Cookbook [http://argouml-stats.tigris.org/documentation/defaulthtml/cookbook/], el FAQ [http://argouml.tigris.org/faqs/users.html], y la Quick Guide [http://argouml.tigris.org/documentation/defaulthtml/quickguide/].

1.3. Visión General del Manual de Usuario

1.3.1. Estructura del Manual Tutorial

Capítulo 2, *Introducción (escribiendose)* proporciona una visión general de OOA&D basada en UML, incluyendo una guia para obtener ArgoUML instalado y funcionando.

Desde Capítulo 4, *Captura de Requerimientos* hasta Capítulo 7, se introduce en cada parte de el diseño de procesos desde la captura de los requerimientos inicial hasta el desarrollo y construcción del proyecto final.

Cuando cada concepto UML es encontrado, su uso es explicado. Su uso dentro de ArgoUML es descrito. Finalmente un caso de estudio es usado para dar ejemplos de los conceptos en uso.

1.3.2. Estructura del Manual de Referencia

Capítulo 8, es una visión general del interfaz de usuario y proporciona un resumen del soporte para los varios tipos de diagrama UML en ArgoUML. Capítulo 10, y Capítulo 11, describen la barra de menu, y cada una de las subventanas de la interfaz de usuario, conocidas como *Paneles*.

Capítulo 15, da detalles de todas las criticas cognitivas dentro del sistema.Finalmente ArgoUML enlazará directamente con este manual cuando se de notificación de las criticas.

Capítulo 16, es una visión general de los artefactos (p.e. las entidades UML que pueden ser colocadas en diagramas) dentro de ArgoUML. Los siguientes capítulos (Capítulo 17, hasta Capítulo 24,) describen los artefactos que pueden ser creados por medio de cada diagrama de ArgoUML, y sus propiedades, tambien como algunos artefactos estandar proporcionados con el sistema.

Se proporciona un completo. Apéndice A, proporciona material para suplementar el estudio de caso usado a lo largo del documento. Apéndice B, y Apéndice C, identifican la información subyacente en UML y las herramientas CASE UML. Apéndice F, es una copia de la GNU Free Documentation License.

Una ambición futura es proporcionar un indice exhaustivo.

1.3.3. Retroalimentación por el Usuario

Por favor, cuentanos que piensas sobre el Manual de Usuario. Tus comentarios nos ayudarán a hacer mejoras. Envia por e-mail tus ideas a la Lista de Correo de Usuarios de ArgoUML [mailto:users@argouml.tigris.org]. En caso de que quisieras añadir en los capitulos sin desarrollar deberias contactar la Lista de Correo de Desarrollador de ArgoUML [mailto:dev@argouml.tigris.org] para comprobar que nadie mas está trabajando en esa parte. Te puedes subscribir a cualquiera de las listas de correo a traves de el Sitio Web de ArgoUML [http://argouml.tigris.org].

1.4. Supuestos

Esta publicación del manual asume que el lector esta ya muy familiarizado con UML. Esto está reflejado en la sobriedad en la descripción de los conceptos UML en el tutorial.

El caso de estudio está descrito, pero aún no totalmente a traves del tutorial. Esto será realizado en futuras publicaciones del manual.

Parte 1. Tutorial

Capítulo 2. Introducción (escribiendose)

Este tutorial te llevará a traves de un tour sobre el uso de ArgoUML para modelar un sistema.

Un proyecto de ATM (automated teller machine; cajero automatico) ha sido escogido como caso de estudio para demostrar los varios aspectos de modelado que ArgoUML ofrece. En subsiguientes secciones vamos a desarrollar el ejemplo de Cajero Automatico en una descripción completa en UML. El tutorial, sin embargo, solo te guiará a traves de parte de ello.

En este punto deberias crear un directorio para contener tu proyecto. Nombra el directorio de forma consistente con el resto de tu sistema de archivos. Deberias nombrar los contenidos y cualquier subdirectorio de forma equivalente por razones que se haran evidentes.

El caso de estudio será un sistema de Cajero Automatico. Tu compañoa es FlyByNight Industries. Tú vas a jugar dos papeles. El de Gestor de Proyecto (Project Manager) y el de Analista Diseñador (Designer Analyst).

No vamos a construir ningún Cajero Automatico fisicamente, por supuesto.

Primero te familiarizaras con el producto y luego iremos a traves de un proceso de analisis y desarrollo para un caso de prueba.

La forma en como tu compañia organiza su trabajo en proyectos está determinada normalmente por asuntos de politicas y demas cosas por el estilo, por tanto, fuera del ámbito de este documento. Iremos dentro de como estructuras el proyecto en sí mismo una vez que ha sido definido.

Capítulo 3. OOA&D basada en UML

En este capitulo, miramos como UML como notación es usado dentro de OOA&D.

3.1. Antecedentes para UML

La *orientacion a Objeto* como concepto ha existido desde la decada de 1960, y como concepto de diseño desde 1972. Sin embargo fué en la decada de 1980 que empezó a desarrollarse como una alternativa creible a el *enfoque funcional* en analisis y diseño. Podemos idientificar un numero de factores.

1. La aparición como corriente dominante de lenguajes de programación OO como SmallTalk y particularmente C++. C++ fué un lenguaje OO pragmatico derivado de C, ampliamente usado por causa de su asociación con Unix.

2. El desarrollo de potentes estaciones de trabajo (workstations), y con ellas la aparición dentro de la corriente dominante de entornos de ventanas para los usuarios. Los Interfaces Graficos de Usuario (Graphical User Interfaces; GUI) tienen una estructura de objetos inherente.

3. Un número de proyectos fallidos muy publicitados, sugiriendo que el enfoque actual no era satisfactorio.

Un numero de investigadores propusieron procesos OOA&D, y con ellos notaciones. Aquellas que alcanzaron cierto exito incluyen Coad-Yourdon, Booch, Rumbaugh OMT, OOSE/Jacobson, Shlaer-Mellor, ROOM (para diseño de tiempo real) y el hibrido Jackson Structured Development.

Durante los tempranos 1990 se hizo claro que estos enfoques tenian muchas buenas ideas, a menudo muy similares. Un gran obstaculo fué la diversidad de notación, significando que los ingenieros tendian a tener familiaridad con una metodologia OOA&D, en lugar de el enfoque en general.

UML fué concevido como una notacion común, que sería de interes para todos los involucrados. El estandar original fué gestionado por Rational Software (www.rational.com [http://www.rational.com], en el cual tres de los investigadores clave en el campo (Booch, Jacobson y Rumbaugh estuvieron involucrados). Produjeron documentos describiendo UML v0.9 y v0.91 durante 1996. El esfuerzo fué tomado ampliamente por la industria a traves del Object Management Group (OMG), ya bien conocido por el estandar CORBA. Una primera propuesta, 1.0 fué publicada al comienzo de 1997, con una mejorada versión 1.1 aprobada ese otoño.

ArgoUML está basado en UML v1.4, la cual fué adoptada por OMG en Marzo del 2000. La versión oficial actual es UML v1.5 fechada en Marzo del 2003, para ser reemplazada pronto por una revisión mayor, UML v2.0, la cual esta en sus etapas finales de estandarización y se espera completa en 2006.

3.2. Procesos Basados en UML para OOA&D

Es importante comprender que UML es una notación para OOA&D. No describe ningñun proceso en particular. Cualquier proceso adoptado, llevará al sistema por un número de fases para ser construido.

1. Captura de Requerimientos. Esto es donde identificamos los requerimientos para el sistema, usando el lenguaje del *dominio del probrema*. En otras palabras, describimos el problema en los terminos del “cliente”.

2. Analisis. Tomamos los requerimientos y empezamos a refundirlos en el lenguaje de la solución—el *dominio de la solution*. En esta etapa, aunque pensando en terminos de una solución, aseguramos

mantener las cosas a un alto nivel, lejos de detalles concretos de una solución especifica-lo que es conocido como *abstracción*.

3. Diseño. Tomamos la especificación de la fase de Analisis y construimos la solución con todo detalle. Nos estamos moviendo de la *abstracción* del problema a su *realización* en terminos concretos.

4. Fase de Construcción. Tomamos el diseño actual y lo escribimos en un lenguaje de programación real. Esto incluye no solo la programación, si no tambien la prueba de que el programa cumple los requerimientos (*verificación*), probando que el programa realmente resuelve el problema del cliente (*validacion*) y escribiendo toda la documentación de usuario.

3.2.1. Tipos de Procesos

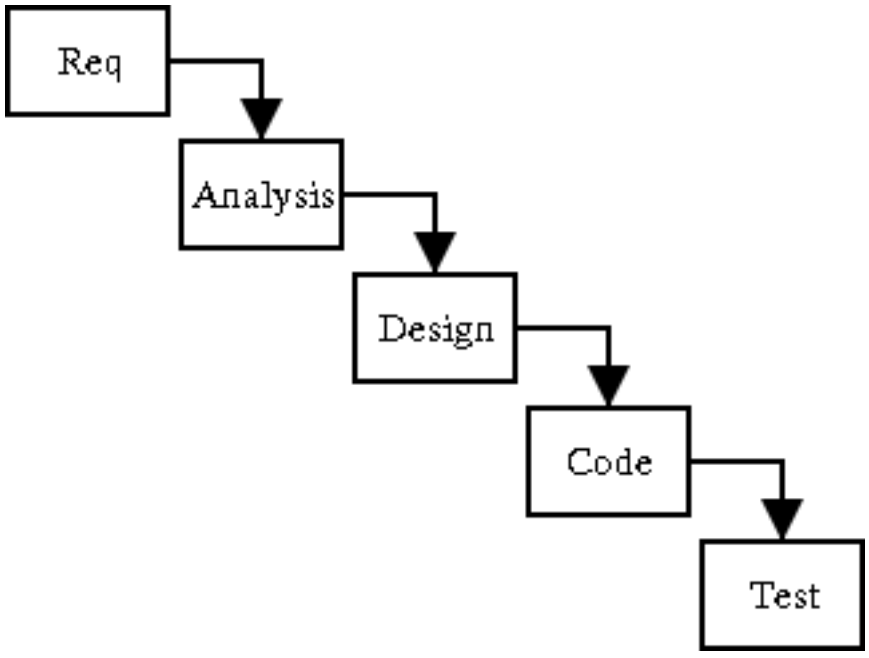
En esta sección miramos a los dos tipos principales de procesos en us por la ingeniería del software. Hay otros, pero son menos ampliamente usados.

En años recientes ha habido tambien un movimiento para reducir el esfuerzo requerido en desarrollar software. Esto ha llevado al desarrollo de un numero de variantes ligeras de procesos (a menudo conocidas como *computacion agil* o *programación extrema*) que son apropiadas para equipos muy pequeños de ingenieros.

3.2.1.1. El Proceso en Cascada

En este proceso, cada etapa del proceso-requerimientos, analisis y construccion (codigo y prueba) es completada antes que la siguiente comienze. Esto se ilustra en Figura 3.1, “ El Proceso en Cascada ”.

Figura 3.1. El Proceso en Cascada



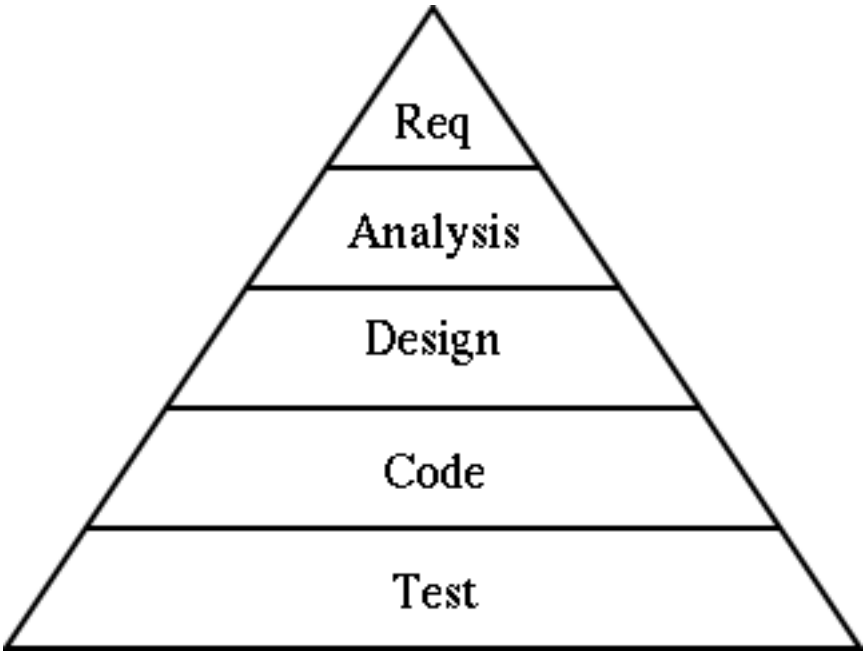
Este es un proceso muy satisfactorio donde los requerimientos están bien diseñados no se espera que cambien, por ejemplo automatizar un sistema manual bien probado.

La debilidad de este enfoque se muestra problemas menos bien definidos. Invaliablemente algunas de las incertidumbres en los requerimientos no serán clarificados hasta bien entrado el analisis y el diseño, o incluso en fases de codificación, requiriendo volver atras para rehacer trabajo.

El peor aspecto de esto, es que no cuentas con codigo que funcione hasta cerca del final del proyecto, y muy a menudo es solo en esta etapa en la que los problemas con los requerimientos originales (por ejemplo con la interfaz de usuario) se hacen visibles.

Esto está exacerbado, por cada etapa sucesiva requiriendo mas esfuerzo que la anterior, así que los costos del decubrimiento de un problema tardío son enormemente caros. Esto esta ilustrado por la piramide en Figura 3.2, " Esfuerzo Involucrado en los Pasos del Proceso en Cascada ".

Figura 3.2. Esfuerzo Involucrado en los Pasos del Proceso en Cascada



El proceso en cascada es probablemente aún el proceso de diseño dominante. Sin embargo debido a sus limitaciones esta cada vez mas siendo sustituido por procesos *iterativos*, particularmente por proyectos donde los requerimientos nos están bien definidos.

3.2.1.2. Procesos de Desarrollo Iterativo

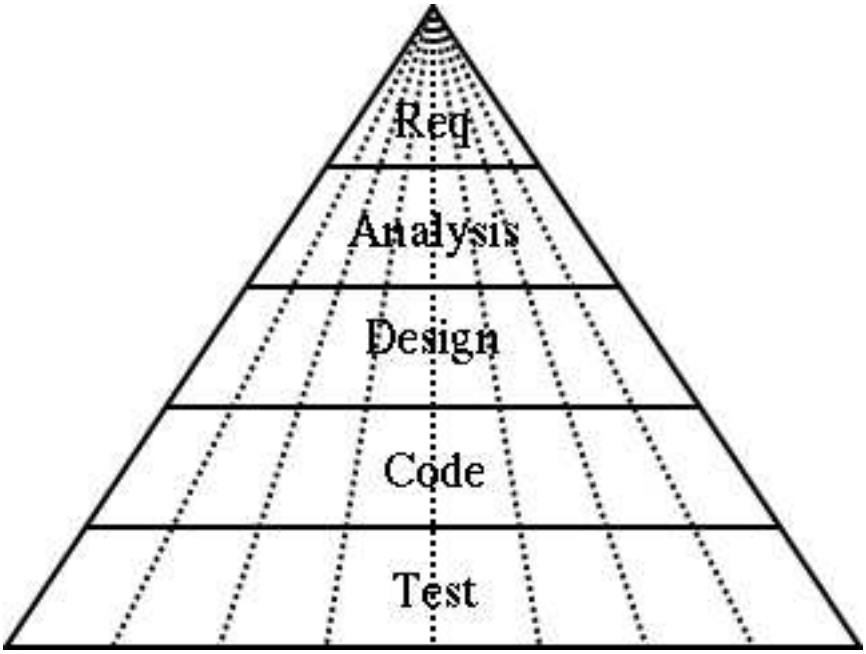
En años recientes un nuevo enfoque ha sido usado, el cual anima a conseguir al menos una parte del codigo funcionando tan pronto como sea posible, para conseguir descubrir problemas antes en el ciclo de desarrollo.

Estos procesos usan unas series de "mini-cascadas", definiendo unos pocos requerimientos (los mas importantes) primero, llevandolos a traves del analisis, diseño y construcción para obtener una version temprana del producto, con funcionalidad limitada, relacionada con los requerimientos mas importantes. La retroalimentación de este código puede ser usada para refinar los requerimientos, apuntar problemas, etc antes de hacer mas trabajo.

El proceso es entonces repetido para requerimientos adiciones para construir un producto con un paso mas en funcionalidad. Otra vez retroalimentación adicional puede ser aplicada a los requerimientos.

El proceso es repetido, hasta que finalmente todos los requerimientos han sido implementados y el producto está completo. Es esta *iteración* lo que da a estos procesos su nombre. Figura 3.3, " Esfuerzo Involucrado en los Pasos de un Proceso Iterativo " muestra como este proceso se compara con la structura piramidal del Proceso en Cascada.

Figura 3.3. Esfuerzo Involucrado en los Pasos de un Proceso Iterativo



El crecimiento en popularidad de los procesos iterativos está estrechamente unido a el crecimiento de OOA&D. Es la encapsulaión limpia de objetos lo que permite a una parte del sistema ser contruida con trozos para el codigo restante claramente definidos.

3.2.1.2.1. El Proceso Racional Unificado

Quizas el Proceso Iterativo mejor conocido es el Proceso Racional Unificado (Rational Unified Process; RUP) de Rational Software (www.rational.com [http://www.rational.com]).

Este proceso reconoce que nuestra vista piramidal de porciones iguales de la cascada no es realista. En la practica las iteraciones tempranas tienden a ser pesadas en los asuntos de requerimientos de cosas (necesitas definir una cantidad razonable incluso para comenzar), mientras las iteraciones posteriores tienen mas esfuerzo en las areas de diseño y construcción.

RUP reconoce que las iteraciones pueden ser agrupadas en un numero de *fases* deacuerdo a su etapa en el proyecto global. Cada fase puede tener una o mas iteraciones.

- En la *fase del principio (inception phase)* las iteraciones tienden a ser pesadas en asuntos de requerimientos/analisis, mientras que cualquier actividad de construcción debe estar limitada a la emulación del diseño dentro de una herramienta CASE.
- En la *fase de elaboración (elaboration phase)* las iteraciones tienden a ser completar la especificación de los requerimientos, y comenzar a centrarse en el analisis y el diseño, y posiblemente la construcción del primer codigo real.

- En la *fase de construcción (construction phase)* los requerimientos y analisis están mas o menos completos, y el esfuerzo de las iteraciones esta mayormente en diseño y construcción.
- Finalmente, en la *fase de desarrollo (deployment phase)* las iteraciones están centradas sobre la actividad de la contrucción, y en particular la prueba del software.

Nota

Debería estar claro que la prueba es una parte integral de todas las fases. Incluso en las fases tempranas los requerimientos y el diseño deberian ser probados, y esto es facilitado por una buena herramienta CASE.

Usaremos un proceso iterativo en este manual, que está ligeramente basado en RUP.

3.2.1.2.2. Tamaño de Iteración

Una buena regla a primera vista es que una iteración debería tomar entre seis y diez semanas para proyectos comerciales tipicos. Mas largo y probablemente habras abarcado demasiados requerimientos para hacerlos de una vez. Ademas pierdes enfoque en tener la siguiente iteración completa. Mas corto y probablemente no has tomado en cuenta suficientes requerimientos para hacer un avance significativo. En este caso la sobrecarga adicional asociada con una iteración puede hacerse un problema.

El número total de iteraciones depende del tamaño del proyecto. Toma el tiempo estimado (trabajando fuera/adivinando que es un tema completo en si mismo), y dividelo en trozos de 8 semanas. La experiencia parece sugerir que las iteraciones se dividiran en una proporcion de alrededor de 1:2:3:3 dentro del estilo RUP de fases de inception, elaboration, construction y deployment. Un proyecto que tiene una gran imprecisión en su especificación (algunos proyectos de investigación avanzada por ejemplo) tenderan a ser mas pesados en sus fases tempranas.

Cuande se construlle un producto por contrato para un cliente el punto final esta bien definido. Sin embargo, cuando se desarrolla un nuevo producto para el mercado, una estrategia que puede se usada es decidir la fecha de lanzamiento del producto, y por tanto la fecha final para completar las labore de ingenieria (algún tiempo antes). El tiempo está entonces dividido en iteraciones, y la cantidad del producto que puede ser construido en el tiempo desarrollado. El processo iterativo es muy efectivo donde el timepo para la comercialización es mas importante que la funcionalidad exacta.

3.2.1.3. Procesos de Desarrollo Recursivo

Muy pocos sistemas software están concevidos como artefactos monoliticos. Están divididos en subsistemas, modulos, etc.

Los procesos de Software son iguales, con partes tempranas del proceso definiendo una estructura de alto nivel, y reaplicando el proceso para partes de la estructura en turnos para definir cada vez mayores detalles.

i.
ii.
iii.
iv.

Por ejemplo, el diseño inicial de un sistema telefonico puede identificar objetos para i) manejar las lineas de telefono, ii) procesar las llamadas, iii) manejar el sistema y iv) facturar al cliente. Los procesos de software pueden entonces ser reaplicados a cada uno de esos cuatro componentes para identificar su diseño.

OOA&D con sus limites claros a los objetos, soporta naturalmente este enfoque. Esta clase de OOA&D con desarrollo recursivo se abrevia a veces como OOA&D/RD.

El desarrollo Recursivo puede ser aplicado igualmente bien a procesos de cascada o iterativos. No es una alternativa a ellos.

3.2.2. Un Proceso de Desarrollo para este Manual

Para el proposito de este manual usaremos un proceso iterativo descendente con desarrollo recursivo, ligeramente semejante a RUP. El caso de estudio nos llevará a traves de la primera iteración, aunque al final de la sección de tutorial del manual miraremos a como el proyecto se desarrollará hasta su finalización.

Dentro de la primera iteración, abordaremos cada uno de las actividades de captura de requerimientos, analisis, diseño y construcción por turno. No todas las partes del procesos están basadas en UML o ArgoUML. Miraremos a que otro material es necesario.

Dentro de este proceso tendremos una oportunidad para ver los varios diagramas UML en uso. El rango completo de diagramas UML y como están soportados está descrito en el manual de referencia (mira Sección 16.8, “ ”).

3.2.2.1. Captura de Requerimientos

Nuestra captura de requerimientos usará el concepto UML de *Casos de Uso*. Empezando con un *Vision Document* veremos como los Casos de Uso pueden ser desarrollados para describir todos los aspectos del comportamiento del sistema en el dominio del problema.

3.2.2.2. Analisis

Durante la etapa de analisis, introduciremos el concepto de UML de *clases* para permitirnos construir una visión de alto nivel de los objetos que conformaran la solución—a veces conocida como *diagrama de concepto*.

Introduciremos el *diagrama de sequencia* y *diagrama de estados* para capturar requerimientos por el comportamiento global del sistema.

Finalmente, tomaremos los Casos de Uso de la etapa de captura de requerimientos, y remoldearlos en el lenguaje del dominio de la solución. Esto ilustrará las ideas UML de *estereotipado* y *realización*.

3.2.2.3. Diseño

Usamos el *diagrama de paquetes* UML para organizar los componentes del proyecto. Luego revisaremos el diagrama de clases, diagrama de secuencia y diagrama de estados, para mostrar como pueden ser usados recursivamente para diseñar la solución completa.

Durante esta parte del proceso, necesitamos desarrollar nuestra arquitectura del sistema, para definir como todos los componentes ajustaran juntos y funcionaran.

Aunque no es estrictamente parte de nuestro proceso, miraremos a como el *diagrama de colaboración* UML puede ser usado como una alternativa para, o complementar el *diagrama de secuencia*. Similarmente miraremos al *diagrama de actividades* UML como una alternativa o complemento para el diagrama de estado.

Finalmente usaremos el *diagrama de despliege* UML para especificar como el sistema será finalmente realizado.

3.2.2.4. Construcción

UML no está realmente afectado con la escritura de codigo. Sin embargo, en esta etapa mostraremos como ArgoUML puede ser usado para generación de codigo.

Tambien miraremos a como el Diagrama de Casos de Uso UML y la Especificación de Casos de Uso son herramientas invalorables para un programa de prueba.

3.3. Por que ArgoUML es Diferente

En la introducción, listamos las cuatro aspectos clave que hacen a ArgoUML diferente: i) hace uso de ideas de psicologia cognitiva, ii) está basado en estandares abiertos; iii) es 100% Java puro; y iv) es un proyecto de codigo abierto.

3.3.1. Psicología Cognitiva

3.3.1.1. Teoría

ArgoUML está particularmente inspirado en tres teorias dentro de la psicología cognitiva: i) reflexión-en-acción, ii) diseño oportunista iii) y comprensión y resolución de problemas.

- *Reflexión-en-Acción*

  Esta teoría observa que los diseñadores de sistemas complejos no conciven un diseño totalmete formado. En su lugar, deben construir un diseño parcial, evaluarlo, reflexionar en el, y revisarlo, hasta que estén listos para extenderlo mas allá.

  Como los desrrolladores trabajan directamete sobre el diseño, sus modelos mentales de la situación del problema mejoran, por lo tanto mejoran sus diseños.

- *Diseño Oportunista*

  Una teoría dentro de la psicología cognitiva sugiere que aunque los diseñadores planean y describen su trabajo de una forma jerarquica ordenada, en realidad, escogen tareas sucesivas basados en el criterio de costo cognitivo.

  Explicado simplemente, los diseñadores no sigue incluso sus propios planes en orden, si no que escogen pasos que son mentalmente menos caros entre las alternativas.

- *Comprensión y Resolución de Problemas*

  Una teoría de visualización de diseño dentro de la psicología cognitiva. La teoría expone que los diseñadores deben cubrir un hueco entre su modelo mental del problema o situación y el modelo formal de una solución o sistema.

  Esta teoría sugiere que los programadores de beneficiaran de:

  1. Representaciones multiples como descomposición sintactica del programa, transiciones de estado, flujo de control, y flujo de datos. Estos permiten al programador identificar mejor elementos y relaciones en el problema y solución y por lo tanto mas facilmente crear un mapeo entre sus modelos de situación y modelos del funcionamiento del sistema.

  2. Aspectos familiares de un modelo de situación, que mejoran las abilidades de los diseñadores para formular soluciones.

3.3.1.2. Aplicación Practica en ArgoUML

ArgoUML implementa estas teorías usando un numero de tecnicas.

1. El diseño de un interfaz de usuario que permite al usuario ver el diseño desde un numero de perspectivas diferentes, y permite al usuario alcanzar objetivos a traves de un numero de rutas alternativas.

2. El uso de procesos ejecutandose en paralelo con la herramienta de diseño, evaluando el diseño actual contra modelos de como un diseño de la “mejor practica” puede funcionar. Estos procesos son conocidos como *criticos de diseño*.

3. El uso de *listas de tareas pendientes (to-do lists)* para comunicar sugerencias desde los criticos de diseño al usuario, ademas de permitir al usuario registrar areas para acciones futuras.

4. El uso de listas de validación, para guiar al usuario a traves de un proceso complejo.

3.3.2. Estandares Abiertos

UML es en si mismo un estandar abierto. ArgoUML sobre todo ha intentado usar estandares abiertos para todas sus interfaces.

La ventaja clave de la adherencia a los estandares abiertos es que ello permite un facil interfuncionamiento entre aplicaciones, y la abilidad de moverse de una aplicación a otra como sea necesario.

3.3.2.1. XML Metadata Interchange (XMI)

XML Metadata Interchange (XMI) es el estandar para guardar los meta-datos que confeccionan un modelo UML particular. En principio esto te permitirá tomar el modelo que has creado en ArgoUML y importarlo dentro de otra herramienta.

Esto claramente tiene ventajas in permitir al UML alcanzar su meta de ser un estandar para la comunicación entre diseñadores.

La realidad no es tan buena. Anteriormente a UML 2.0 el archivo XMI no incluye información sobre la representación grafica de los modelos, así que el diseño del diagrama está perdido. ArgoUML rodea este problema guardando la información grafica separada del modelo (mira Sección 3.4.3.1, “ Cargando y Guardando ”).

3.3.2.2. Formatos Graficos - EPS, GIF, PGML, PNG, PS, SVG

- *Encapsulated PostScript (EPS)* [http://en.wikipedia.org/wiki/Encapsulated_PostScript] es un archivo PostScript que satisface restricciones adicionales. Estas restricciones estan previstas para hacer mas facil para el software incrustar un archivo EPS dentro de otro documento PostScript.

- *Graphics Interchange Format (GIF)* [http://en.wikipedia.org/wiki/GIF] es un formato cubierto por patente, aunque las patentes se agotaran en Agosto del 2006.

- *Precision Graphics Markup Language (PGML)* [http://en.wikipedia.org/wiki/PGML] es un lenguaje basado en XML para representar graficos vectoriales. Fué un borrador de la W3C, pero no fué adoptado como recomendación. PGML y VML, otro lenguaje basado en XML para graficos vectoriales, fueron luego fusionados y mejorados para crear juntos el formato SVG.

- *Portable Network Graphics (PNG)* [http://en.wikipedia.org/wiki/PNG] es un estandar de ISO/IEC (15948:2004) y es tambien una recomendación de la W3C. PNG es un formato de imagen de mapa de bits que emplea compresión de baja perdida. PNG fué creado para mejorar y sustituir el formato GIF con un formato de archivos de imagen que no requiriera de una licencia de patense para ser usado. PNG es oficialmente pronunciado "ping" pero a menudo es simplemente deletreado — probablemente para evitar confusión con la utilidad de red ping. PNG está soportado por la librería de referencia libpng, una librería independiente de la plataforma que contiene funciones C para el manejo de imagenes PNG.

- *PostScript (PS)* [http://en.wikipedia.org/wiki/PostScript/] es un lenguaje de descripción de pagina y lenguaje de programación usado primeramente en la areas de publicación asistida por ordenador y electronica.

- *Scalable Vector Graphics (SVG)* [http://en.wikipedia.org/wiki/Scalable_Vector_Graphics] es un lenguaje de marcado XML para describir graficos vectoriales bidimensionales, estaticos y animados, y de forma declarativa o mediante scrip. Es un estandar abierto creado por el World Wide Web Consortium. El uso de SVG en la web esta en sus primeros pasos. Hay un gran problema en la inercia debida al largo tiempo de uso de formatos basados en mapas de bits y otros formatos como Macromedia Flash or Java applets, pero ademas el soporte por navegadores web es todavía desigual, con soporte nativo en Opera y Firefox, pero Safari y Internet Explorer necesitan un plugin. Mira PGML mas arriba.

3.3.2.3. Object Constraint Language (OCL)

Object Constraint Language (OCL) [[http://en.wikipedia.org/wiki/Object_Constraint_Language] es un lenguaje declarativo para describir reglas que se aplican a modelos UML. Fué desarrollado en IBM y ahora es parte del estandar UML. Inicialmente OCL fué solo una especificación formal de una extensión de lenguaje para UML. OCL puede ahora ser usado con cualquier metamodelo compatible con Meta-Object Facility (MOF), incluyendo UML. El Object Constraint Language es un lenguaje de texto preciso que proporciona limitación y expresiones de acceso a objeto en cualquier modelo MOF o metamodelo que de otra forma no puede ser expresado por notación diagramatical.

3.3.3. 100% Java Puro

Java fué concevido como un lenguaje interpretado. No tenía un compilador para producir codigo para cuaquier maquina particular. Compila codigo para su propio sistema, la *Maquina Virtual Java (Java Virtual Machine; JVM).*

Escribir un interprete para una JVM es mucho mas facil que escribir un compilador, y estas maquinas virtuales están ahora incorporadas dentro de casi todo Navegador Web. Como resultado, la mayoria de las maquinas pueden ejecutar Java, sin trabajo adicional.

(En caso de que te preguntes porque todos los lenguajes no son como este, es por que los lenguajes interpretados tienden a ser mas lentos que los compilados. Sin embargo, con el alto rendimiento de los modernos PCs, la ganancia en portabilidad merece la pena para muchas aplicaciones. Mas aún, las modernas caches multinivel pueden suponer que los lenguajes interpretados, que producen un codigo mas denso, pueden realmente no ser tan lentas de todas formas.)

Mediante la elección de escribir ArgoUML en Java puro, se hace inmediatamete disponible para el mayor numero de usuarios con la minima cantidad de esfuerzo.

3.3.4. Codigo Abierto

ArgoUML es un proyecto de *código abierto*. Esto significa que cualquiera puede tener una copia gratis del código fuente, cambiarlo, usarlo para nuevos propositos y cosas así. La única (gran) obligación es

que tú pases tu código de la misma forma a otros. La naturaleza precisa de que puedes hacer y que no puede varia de un proyecto a otro, pero el principio es el mismo.

La ventaja es que un proyecto pequeño como ArgoUML subitamente se abre a una gran cantidad de ayuda adicional de aquellos que pueden indagar en sus ideas sobre como el programa puede ser mejorado. En cualquier momento pueden ser 10, 15, 20 o mas personas haciendo contribuciones significantes a ArgoUML. Hacer esto comercialmente costaría mas de 1 millon de $ al año.

No es solo un espiritu de puro altruismo. Contribuir en un camino para aprender “con las manos en la masa” sobre software puntero. Es una forma de adquirir visibilidad (mas de 100,000 personas han descargado ArgoUML hasta la primavera de 2001). Esto es una gran cantidad de buena esperiencia en suma ¡y muchos empleadores potenciales están viendote!

Y es genial para el ego!

El Código Abierto no excluye ganar dinero. Gentleware www.gentleware.com [http://www.gentleware.com] vende una versión comercial de ArgoUML, Poseidon. Su propuesta de valor no es un trozo de códogo privativo. Es el refinamiento comercial y el soporte para librarse de riesgos usando ArgoUML en un desarrollo commercial, permitiendo a los clientes tomar ventaja de la tecnología puntera de ArgoUML.

3.4. Fundamentos de ArgoUML

El objeto de esta sección es ponerte en marcha con ArgoUML. Te llevará a traves de obtener el código y conseguirlo ejecutar.

3.4.1. Empezando

3.4.1.1. Requerimientos del Sistema

Puesto que ArgoUML está escrito en Java puro 100%, debería funcionar en cualquier maquina con Java instalado. Es necesaria una versión 1.4 o posterior de Java. Puedes tenerlo disponible, pero si no, puede ser descargada gratis de www.java.com [http://www.java.com]. Ten en cuenta que solo necesitas el Java Runtime Environment (JRE), no hay necesidad de descargar el Java Development Kit (JDK) completo.

ArgoUML necesita una cantidad razonable de recursos. Un PC con 200MHz de procesador, 64Mb de RAM y 10Mb de espacio disponible en un disco duro debería ser adecuado. Descarga el código de la sección de Descargas de sitio web del proyecto argouml.tigris.org [http://argouml.tigris.org]. Escoje la versión que se ajusta a tus necesidades como se describe en la siguiente sección.

3.4.1.2. Opciones de Descarga

Tienes tres opciones para obtener ArgoUML.

1. Ejecutar ArgoUML directamente desde el Sitio Web usando Java Web Start. Esta el la opción mas facil.

2. Descargar el código binario ejecutable. Esta es la opción correcta si pretendes usar ArgoUML regularmente y no es muy dificil.

3.
4. Descargar el código fuente usando CVS y compilar tu propia versión. Escoge esta opción si quieres mirar el funcionamiento interno de ArgoUML, o quieres unirte como desarrollador. Esta opción requiere el JDK completo (mira Sección 3.4.1.1, “ Requerimientos del Sistema ”).

Las tres opciones están libremente disponibles a traves del sitio web del proyecto, argouml.tigris.org [http://argouml.tigris.org].

3.4.1.3. ArgoUML Usando Java Web Start

Hay dos pasos para esto.

1. Instalar Java Web Start en tu maquina. Está disponible en java.sun.com/products/javawebstart [http://java.sun.com/products/javawebstart], o a traves del enlace `Java Web Start` en la pagina web [http://argouml.tigris.org] de ArgoUML.

2. Haz click en el enlace `Launch latest stable release` de la pagina web [http://argouml.tigris.org] de ArgoUML.

Java Web Start descargará ArgoUML, lo cacheará y lo iniciará por primera vez, luego en subsiguientes inicios, comprueba si ArgoUML está actualizado y solo descarga cualquier parte actualizada y luego lo inicia. La pagina web [http://argouml.tigris.org] de ArgoUML tambien proporciona detalles iniciando ArgoUML desde la consola de Java Web Start.

3.4.1.4.

3.4.1.5. Descargando el Binario Ejecutable

Si escoges descargar el binario ejecutable, tendras la elección de descargar la ultima versión estable del código (la cual será mas fiable, pero no tiene todas las ultimas caracteristicas), o la versión actual (la cual será menos fiable, pero tiene mas caracteristicas). Escoge deacuerdo con tu situación.

ArgoUML está disponible en formatos `.zip` o `tar.gz`. Escoge el primero si eres un usuario de Microsoft Windows, y el ultimo si estás ejecutando alguna variedad de Unix. Desempaquetandolo como sigue.

- En Windows. Descomprime el archivo `.zip` con WinZip, o en ultimas versiones of Windows (ME, XP) copia los archivos fuera de la carpeta comprimida y ponlos en un directorio de tu elección.

- En Unix. Usa GNU tar para descomprimir y desempaquetar los archivos en un directorio de tu elección `tar zxvf <file>.tar.gz`. Si tienes una versión antigual de tar, la opción `z` puede no estar disponible, así que usa `gunzip < file.tar.gz | tar xvf -`.

Deberías tener un directorio conteniendo un numero de archivos `.jar` y un `README.txt`.

3.4.1.6. Problemas Descargando

Si te quedas completamente atascado y no tienes asistencia local, intenta el sitio web, particularmente el FAQ [http://argouml.tigris.org/faqs/users.html]. Si esto aún no resuelve el problema, intentalo en la lista de correo de usuarios de ArgoUML.

Te puedes subscribir a traves de la sección de listas de correo del sitio web del proyecto argouml.tigris.org [http://argouml.tigris.org], o envia un mensaje vacio a users@argouml.org [mailto:users@argouml.org] con el asunto `subscribe` (en ingles).

Puedes entonces enviar tu problema a users@argouml.org [mailto:users@argouml.org] (en ingles) and ver como otros usuarios son capaces de ayudar.

La lista de correo de usuarios es una excelente introducción a la actividad vital del poryecto. Si quieres

estar mas involucrado hay listas de correo adicionales que cubren el desarrollo del producto y problemas en la versión actual y futuras.

3.4.1.7. Ejecutando ArgoUML

Como ejecutar ArgoUML depende de si usas Microsoft Windows o alguna variedad de Unix.

- En Windows. Inicia una interfaz MSDOS por ejemplo usando Inicio/Ejecutar con “command” en el texto texto de la ventana. En la ventana, colocate en el directorio que contiene tus archivos de ArgoUML y teclea `java -jar argouml.jar`. Este metodo tiene la ventaja de que la información de progreso y depuración es visible en la ventana DOS. De forma alternativa crea un archivo de proceso por lotes (.bat) conteniendo el comando anterior, con un acceso directo a el en el Escritorio. El archivo bat debería terminar con una sentencia de "pause" en caso de que se cree información de depuración durante la ejecución. En algunos sistemas, simplemente haciendo doble click en el archivo `argouml.jar` funciona. En otros hacer esto inicia un programa de compresion de archivos. Acude a las instrucciones de tu sistema operativo o busca ayuda para determinar como configurar esto.
- En Unix. Inicia una ventana de terminal y teclea `java -jar argouml.jar`

3.4.1.8. Problemas Ejecutando ArgoUML

Es inusual encontrar problemas si has tenido una descarga exitosa. Si no puedes resolver el problema, prueba la lista de correo de usuario (mira Sección 3.4.1.6, “ Problemas Descargando ”).

- JRE equivocada. El problema mas común es no tener una versión de Java Runtime Environment suficientemente moderna (debe ser 1.4 o posterior).
- Lenguaje equivocado. Si el producto aparece en un idioma que no sabes o no quieres leer, vete a el segundo item de menu por la izquierda en la parte superio de la pantalla. Selecciona la entrada de menu mas baja en la lista desplegada. Figura 3.5, “ Estableciendo Idioma en el Panel de Apariencia ” muestra esto en Ruso. Luego haz click en la segunda solapa por abajo en la columna de solapas de la izquierda. Despliega la lista como se muestra en Figura 3.5, “ Estableciendo Idioma en el Panel de Apariencia ”. y selecciona un idioma. Puedes ver que los idiomas esta listados en ellos mismos. El idioma seleccionado en la muestra es Alemán donde la palabra para “Alemán” es “Deutsch”. Tendras que salir de ArgoUML y reiniciarlo para que los cambios hagan efecto. Usa el boton X de arriba a la derecha.

Figura 3.4. Encontrando el Asistente de Configuración

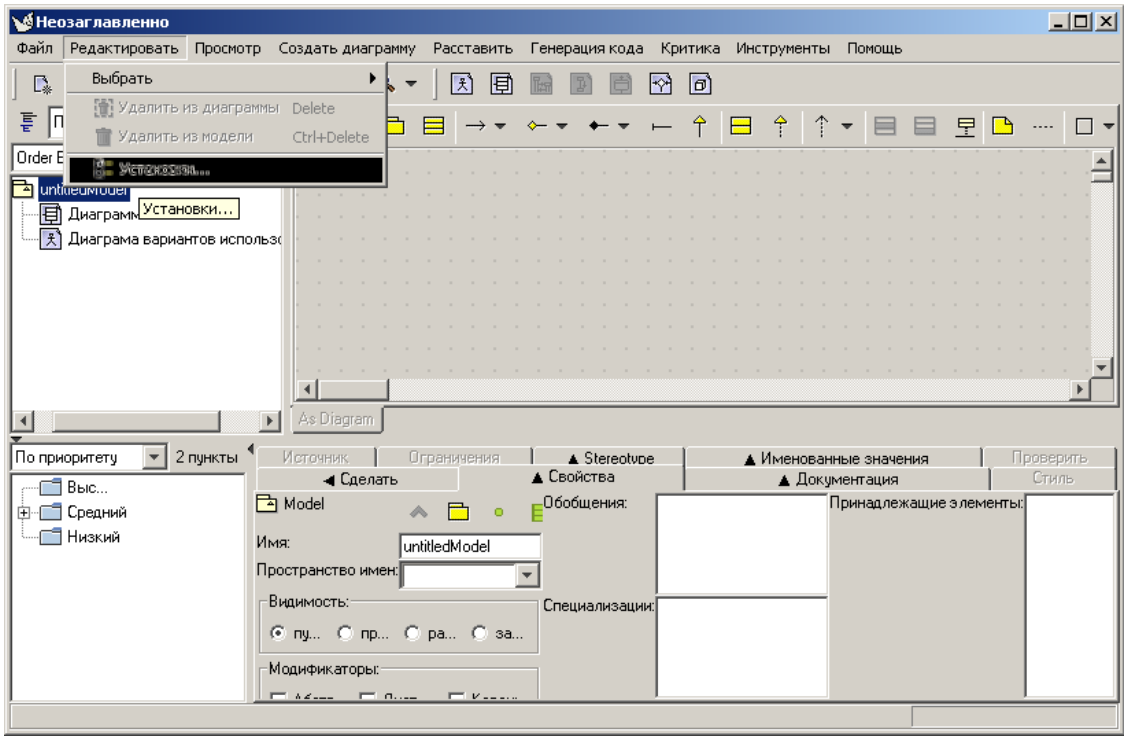
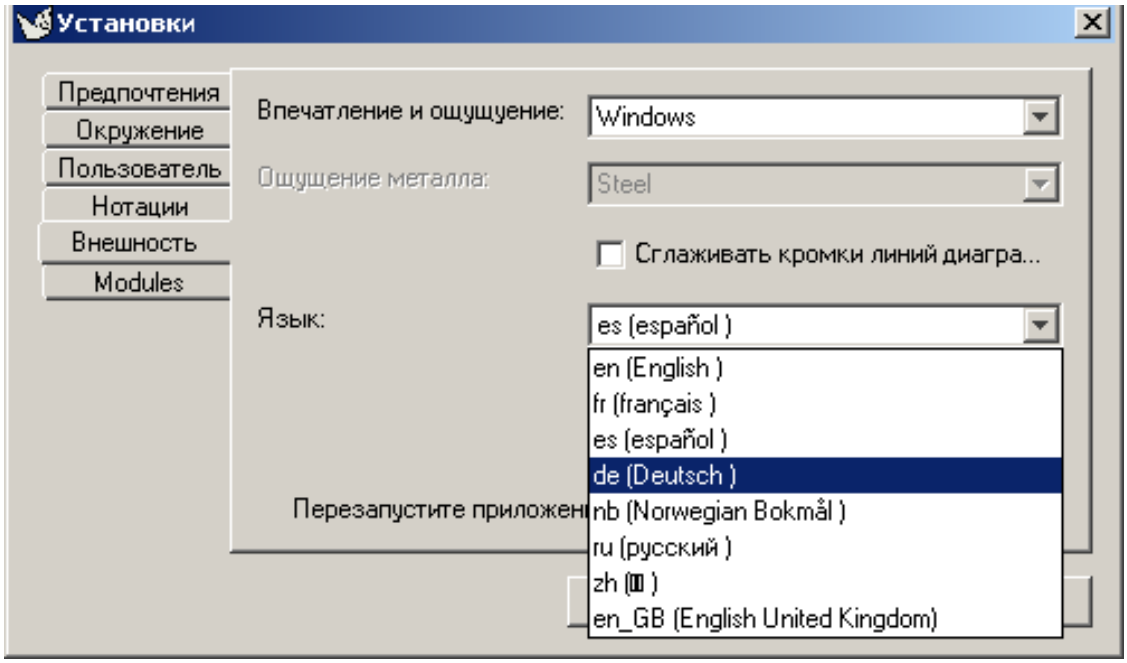


Figura 3.5. Estableciendo Idioma en el Panel de Apariencia



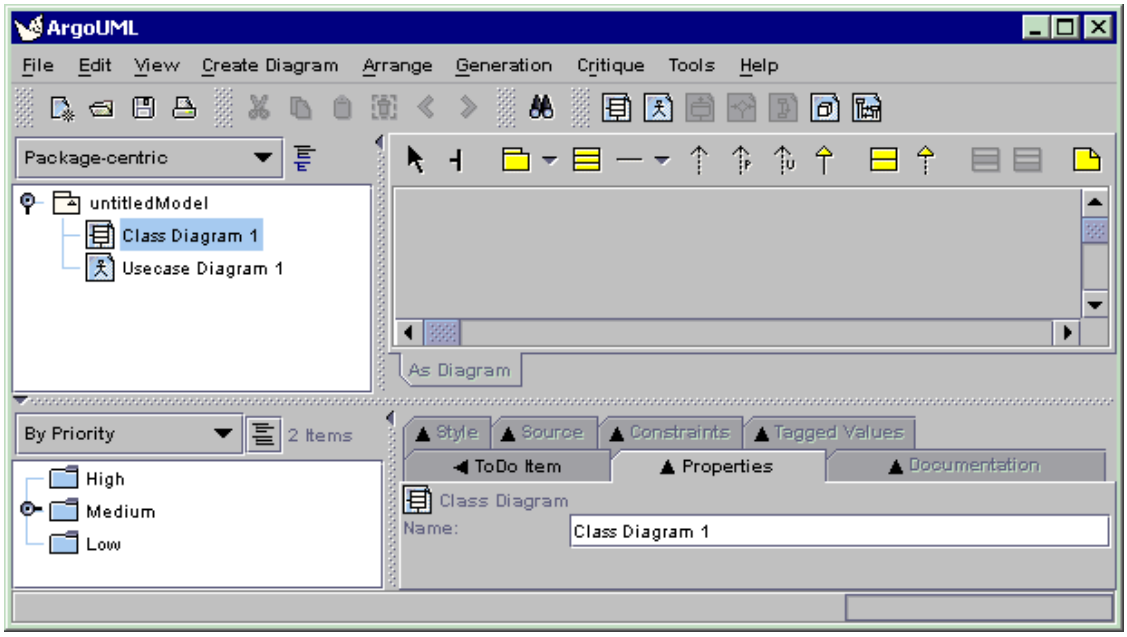
3.4.2. El Interfaz de Usuario de ArgoUML

Antes de empezar el Caso de Estudio, necesitas familiarizarte con el interfaz de usuario. Comienza leyendo la introducción a la Referencía de la Interfaz de Usuario. Mira Capítulo 8, .

Mientras pasas a traves de este tutorial se te dira que hacer, y cuando hacerlo pero como hacerlo a menudo será dejado a la Referencia de la Interfaz de Usuario. No es necesario en este punto leerse toda la Referencia, pero deberías echarle un vistazo hasta familiarizarte con como encontrar cosas en ella. Todo intento será hecho para dirigirte a la parte apropiada de la Referencia en donde se aplican esos puntos en el tutorial.

Figura 3.6, “ Ventana inicial de ArgoUML ”, muestra la ventana principal de ArgoUML como aparece cuando es entra por primera vez en ArgoUML.

Figura 3.6. Ventana inicial de ArgoUML



Agarra la barra divisoria vertical y muevela atras y adelante. Agarra la barra divisoria horizontal y muevela arriba y abajo. Juega un poco por ahí con las pequeñas flechas a la izquierda o arriba de las barras divisorias. Mira Sección 8.3, “ ”;.

3.4.2.1. La Barra de Menú y Barras de Herramientas

La barra de menú y las barras de herramientas dan acceso a todas las caracteristicas principales de ArgoUML. Así las opciones de menú, convencionales y de la barra de herramientas que no están disponibles estan sombreadas y los items de menú que invocan un cuadro de dialogo están seguidos por una elipsis (...).

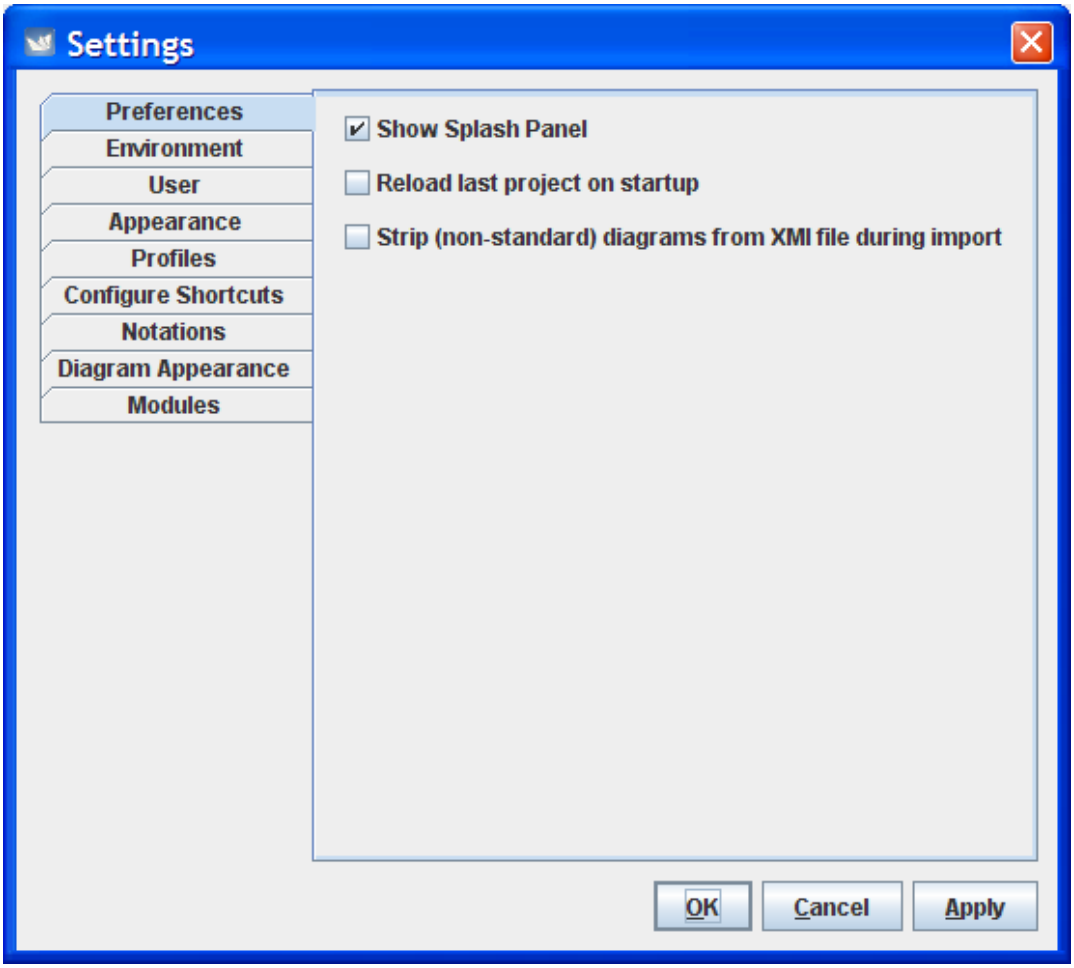
Menú *Archivo*. Te permite crear un nuevo proyecto, guardar y abrir poryectos, importar fuentes desde cualquier sitio, cargar y guardar el modelo en y desde una base de datos, imprimir el modelo, guardar los gráficos del modelo, guardar la configuración del modelo y salir de ArgoUML

1.
2.
3.
4.
5.
6.

Menú *Editar*. Te permite seleccionar uno o mas elementos UML en un diagrama, deshacer y rehacer ediciones, eliminar elementos de diagramas o el modelo completo, vaciar la papelera y cambiar la configuración.

1.
   -
   -
   -
   -
   -
   -
   -
   -
2.
3.

Figura 3.7.



4.
5.
6.
7.
8.

9.
10.
11.

Menú *Visualizar*. Te permite cambiar entre diagramas, encontras objetos en el modelo, hacer zoom en un diagrama, seleccionar una representación de diagrama particular (aunque en este momento solo hay una), seleccionar una etiqueta particular en el menú de detalles, ajustar la rejilla, ver botones en una selección, y cambiar entre notación UML y Java.

1.
2.
3.
4.
5.
6.
7.

Menú *Crear Diagrama*. Te permite crear cualquiera de los siete tipos de diagrama UML (clases, casos de uso, estados, actividad, colaboración, desarrollo y secuencia) soportados por ArgoUML.

Los diagramas de estado y actividad solo pueden ser creados cuando una clase o actor es seleccionado, incluso las entradas relevantes del menú estarán sombreadas si esto no se ha realizado.

Menú *Colocar (Arreglo)*. Te permite alinear, distribuir, reordenar y desplazar objetos en un diagrama y establecer la estrategia de distribución del diagrama.

Menú *Generar*. Te permite generar código Java para las clases seleccionadas o para todas las clases.

Menú *Crítica*. Te permite cambiar el estado de la función de auto-crítica entre activado y desactivado, establecer el nivel de importancia de los problemas de diseño y metas de diseño y inspeccionar las críticas disponibles.

Menú *Herramientas*. Este menú está sombreado permanentemente a menos que halla alguna herramienta disponible en tu versión de ArgoUML.

Menú *Ayuda*. Este menú te da acceso a detalles de aquellos que crearon el sistema, y donde se puede encontrar ayuda adicional.

Barra de Herramientas de Archivo. Esta barra de herramientas contiene algunas de las herramientas del menú Archivo.

Barra de Herramientas de Edición. Esta barra de herramientas contiene algunas de las herramientas de menú Edición.

Barra de Herramientas de Visualizar. Esta barra de herramientas contiene algunas de las herramientas del menú Visualizar.

Barra de Herramientas de Crear Diagrama. Esta barra de herramientas contiene algunas de las herramientas del menú Crear Diagrama.

3.4.2.2. El Panel Explorador

En este momento deberías tomarte tiempo para leer Capítulo 11, . No hay mucho que puedas hacer en este punto con el Panel Explorador ya que no hay nada in el salvo la raiz del arbol (actualmente "untitledModel") y dos diagramas vacíos. Sin embargo, el Panel Explorador es fundamental para casi todo lo que haces y volveremos atras a él una y otra vez en lo sucesivo.

Hay un control de expansión y contracción delante del símbolo del paquete para “untitledModel” en el

Panel Explorador y el símbolo de paquete para “Medium” en el Panel de Tareas Pendientes. Haz click en esos controles y observa que esos paneles son tres widgets que se comportan de forma muy parecida a como se esperaría que hicieran. El control de expansión o contracción es un signo de mas (+)/menos (-) o un pomo dirigido a la derecha o abajo dependiendo del look and feel que has escogido como apariencia.

En este punto deberías probar las varias opciones disponibles para un look and feel (apariencia). Usaste el editor que establece el look and feel cuando estabas seleccionando el idioma, sin embargo, solo lo viste en Ruso. Si miras la versión Española (Inglesa en la imagen) Sección 10.4.5.4, “ ” veras que la combobox de la zona mas alta es para seleccionar el look and feel. Cuando el panel se abre por primera vez la caja contiene el valor actual. Selecciona otra, sal de ArgoUML y reinicialo.

Seleciona alternativamente Diagrama de clase 1 y Diagrama Use Case 1 observando que el panel de detalle cambia siguiendo los objetos seleccionados en el Explorador. El panel de detalle está descrito en el Capitulo 12. No es necesario leer el capitulo 12 en este punto, pero tampoco te hara daño.

3.4.2.3. El Panel de Edición

Nota

- Taréa de Lectura.
- Pasa a traves de un par de cambios.
- Añade algunas cosas.
- Elimina algunas cosas.
- Redimensiona cosas.
- Selecciona cosas con arrastrar y soltar.
- Seleciona cosas con click y ctrl click.
- Edita nombres integrados.
- Elimina "images/tutorial/editoverview.gif" del sistema de archivos.

3.4.2.4. El Panel de Detalles

Nota

- Taréa de Lectura.
- Item Taréas Pendientes. Trata las diferencias con otras etiquetas sobre localizaciones de items seleccionados. Mantiene detalles para tratarlos en el Panel de Taréas Pendientes.
- Propiedades,
- Documentación,

- Presentación,
- Fuente,
- Constantes,
- Estereotipo,
- Valores Etiquetados,
- Lista de Validación.

3.4.2.5. El Panel de Taréas Pendientes

Nota

- Taréa de Lectura.
- Describe prioridades.
- Resolver items.
- Relationa a una etiqueta de Item Pendiente en el panel de detalles.

3.4.2.6. Dibujando Diagramas

En general los diagramas son dibujados usando la barra de herramientas del panel de edición para seleccionar el objeto deseado y haciendo click en el diagrama en la posición requerida . Esa sección tambien explica el uso del ratón para redimensionar objetos.

Los objetos que ya están en el modelo, pero no en un diagrama, pueden ser añadidos a un diagrama seleccionando el objeto en el explorador, usando `Agregar al Diagrama` del menú desplegado (boton 2) sobre ese objeto, y entonces haciendo click button 1 en la posición deseada en el diagrama.

Ademas de objetos UML, la barra de herramaientas del panel Edición proporciona para los objetos de dibujados generales (rectangulos, circulos, lineas, poligonos, curvas, texto) formas de proporcionar información suplementaria para los diagramas.

3.4.2.6.1. Moviendo Elementos de Diagrama

Hay muchas maneras para mover elementos de diagrama.

3.4.2.6.1.1. Usando Teclas del Ratón

Seleciona los elementos que quieres mover. Presionando la tecla Ctrl mientras seleccionas puedes seleccionar muchos elementos para mover el mismo tiempo.

Ahora presiona tus teclas de flecha. Los elementos seleccionados se mueven un poco con cada tecleo.

Si ademas mantienes presionada la tecla Mayusculas (Shift), se moveran un poco mas rápido.

3.4.2.6.1.2. Usando la Barra de Herramientas del Panel de Edición

Haz click en el boton escoba en la barra de herramientas. Mueve tu ratón al panel de diagrama, haz click derecho y mantenlo. Ahora mover tu ratón alineará los elementos.

3.4.2.6.2. Colocando Elementos

El elemento de menú `Colocar (Arreglo)` te permite alinear, agrupar, o desplazar elementos.

3.4.2.7. Trabajando con Proyectos

3.4.2.7.1. La Ventana de Inicio

Figura 3.6, " Ventana inicial de ArgoUML " muestra la ventana principal de ArgoUML como aparece justo despues de iniciar

El area de cliente de la ventana principal, debajo del menú y la barra de herramientas, está subdividida en cuatro paneles. Empezando con el panel superior mas a la izquierda, y funcionando continuamente, puedes ver el Explorador, mostrando una vista de arbol de tu modelo UML, el Panel de Edición con su barra de herramientas, dos barras de desplazamiento y un area de gris de dibujado, el Panel de Detalles con la solapa de Tareas Pendientes seleccionada, y el Panel de Tareas Pendientes con una vista de arbol de los tareas pendientes, ordenadas de distintas maneras seleccionadas via la lista desplegable en lo alto del panel.

Cada vez que ArgoUML es iniciado sin un archivo de proyecto como argumento, un nuevo proyecto en blanco es creado. Este proyecto contiene un modelo llamado `untitledModel`. Este modelo contiene un Diagrama de Clases en blanco, llamado `Diagrama de clase 1`, y un Diagrama de Casos de Uso llamado `Diagrama use case 1`.

El modelo y los dos diagramas vacios se pueden ver en el explorador, que es la herramienta principal para navegar a traves de tu modelo.

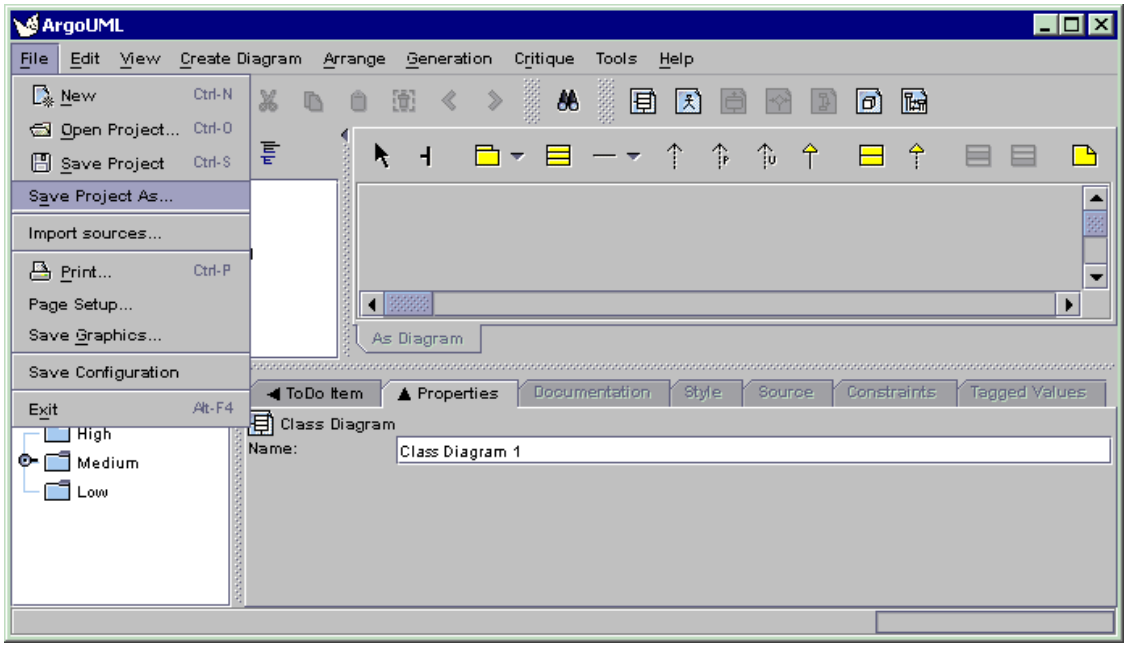
Asumamos por un momento que este el el punto donde quieres empezar a modelar un nuevo sistema de compras. Tú quieres darle el nombre "purchasingmodel" a tu modelo, y quieres almacenarlo en un archivo llamado `FirstProject`.

3.4.2.7.2. Guardando un Proyecto - El Menú Archivo

Por ahora ArgoUML guarda diagramas usando un estandar propuesto recientemente, *Precision Graphics Markup Language (PGML)*. Sin embargo, tiene la opción de exportar datos gráficos como SVG para aquellos que quieran hacer uso de ello. Cuando ArgoUML soporte UML 2.0, almacenará diagramas usando el UML 2.0 Diagram Interchange format.

Primero, salva el modelo en su estado (vacio y sin nombre) actual. En la barra de menú, haz click en `Archivo`, luego en `Guardar Proyecto como...` como se muestra en Figura 3.8, " Invocando `Guardar Proyecto como...` ".

Figura 3.8. Invocando `Guardar Proyecto como...`



Por favor, date cuenta que el menu Archivo contiene las opciones usuales para crear un nuevo proyecto, para abrir un proyecto existente, para guardar un proyector bajo un nuevo nombre, para imprimir del diagrama actualmente mostrado, para guardar el diagrama mostrado actualmente como archivo, y para salir del programa.

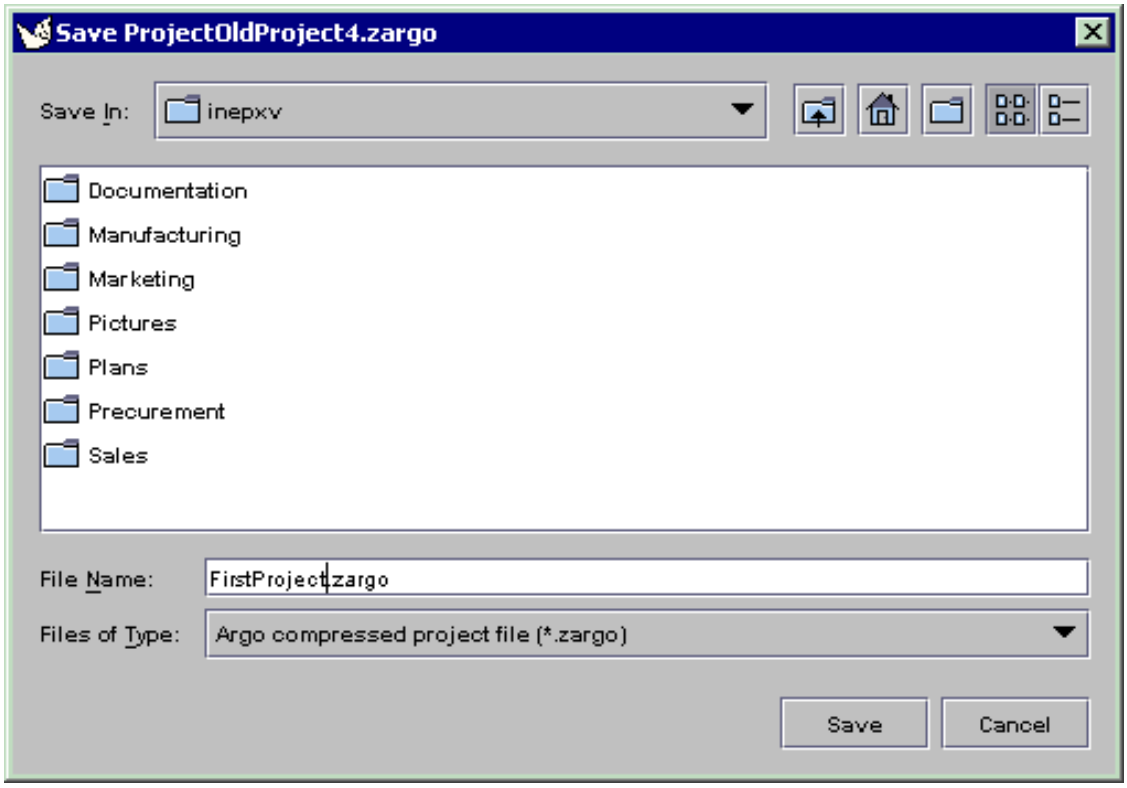
En la versión actual, ArgoUML solo puede contener un proyecto activo al mismo tiempo. Ademas, un poryecto solo puede contener un modelo UML. Puesto que un modelo UML puede contener un número ilimitado de elementos y diagramas, esto no debería presentar ninguna limitación seria, incluso modelando sistemas bastante grandes y complejos.

En la versión actual, ArgoUML solo puede contener un proyecto activo al mismo tiempo. Ademas, un poryecto solo puede contener un modelo UML. Puesto que un modelo UML puede contener un número ilimitado de elementos y diagramas, esto no debería presentar ninguna limitación seria, incluso modelando sistemas bastante grandes y complejos.

3.4.2.7.3. El Diálogo de Elección de Archivo

Pero volvamos a guardar nuestro proyecto. Despues de hacer click en el comando de menú `Guardar Proyecto como...`, obtenemos el dialogo de elección de archivo para introducir el nomvre del archivo que deseamos usar como se muestra en Figura 3.9, “ Diálogo de Elección de Archivo ”.

Figura 3.9. Diálogo de Elección de Archivo



Este es un FileChooser Java estandar. Vayamos sobre ello con algún detalle.

La caracteristica principal y destacada, es la lista de carpetas con barras de desplazamiento en el centro del diálogo. Usando la barra de desplazamiento en la derecha, puedes moverte arriba y abajo en la lista de carpetas contenida dentro de la carpeta actualmente seleccionada. Si puedes desplazarte o no, depende de la cantidad de archivos y carpetas mostradas y ademas como están mostradas. Si todo ajusta, entonces la ventana no es desplazable como se ve en la ilustración.

Haciendo Doble Click en una de las carpetas mostradas te introduce hace navegar dentro de esa carpeta, permitiendote navegar rapidamante entre la jerarquia de carpetas de tu disco duro.

Date cuenta que solo los nombres de carpetas, y no nombres de archivo son mostrados en el area navegable. De echo, el diálogo está actualmente dispuesto deacuerdo para mostrar solo archivos de proyecto ArgoUML con la extension `.zargo`, como puede verse en el control de la parte baja de la ventana etiquetado como `Archivos de Tipo:`.

Ten en cuenta tambien que el nombre de carpeta seleccionado actualmente es mostrado en el control desplegable de la parte alta de la ventana etiquetado como `Buscar en:`. Un unico click en una carpeta dentro del area navegable selecciona la carpeta en la pantalla pero no selecciona la carpeta para guardar.

En la parte superior del diálogo, sobre el area navegable de selección de carpetás, hay algunas herramientas mas para navegar entre carpetas.

- 

El control desplegable de Carpeta. Haciendo click en la flecha hacia abajo se muestra una vista en arbol de la jerarquia de la carpeta, permitiendote navegar rapidamente en la jerarquia, y al mismo tiempo determinar rapidamente donde nos encontramos posicionados en ella.

-  El icono de Subir un Nivel. Haciendo click en este icono nos desplazaremos a la carpeta padre de la carpeta actual.
-  El icono de Principal. Haciendo click en este icono nos desplazaremos a nuestro directorio home.
-  El icono Nueva Carpeta. Haciendo click en este icono crearemos una nueva carpeta llamada "Carpeta nueva" bajo la carpeta actual. Despues de que la carpeta está creada puedes seleccionarla y hacer click en su nombre para cambiarle el nombre a tu elección.
-  El icono de Presentación de Carpetas.

De acuerdo, ahora navegamos al directorio donde queremos guardar nuestro proyecto ArgoUML, rellena el `Nombre de Archivo:` con un nombre apropiado, como “PrimerProyecto” y haz click en el boton `Guardar`.

Ahora tienes un poryecto activo llamado `PrimerProyecto`, conectado al archivo `PrimerProyecto.zargo`.

3.4.3. Salida

3.4.3.1. Cargando y Guardando

3.4.3.1.1. Guardar archivos XMI en ArgoUML

ArgoUML guarda la información de diagrama en un archivo PGML (con extensión `.pgml`, la información del modelo en un archivo XMI (con extensión `.xmi` y la información sobre el proyecto en un archivo con extensión `.argo`. Mira Sección 3.4.3.2.2, “ Precision Graphics Markup Language (PGML) ” y Sección 3.4.3.3, “XMI” para ver mas acerca de PGML and XMI respectivamente.

Todos estos son luego comprimidos en zip en un archivo con extensión `.zargo`. Puedes extraer facilmente el archivo `.xmi` del archivo `.zargo` usando cualquier aplicación generica ZIP. Intentalo y mira dentro de la magia de.

Aviso

Seguramente hacer doble click lanzara una `utilidad ZIP`, si una está instalada, y NO Argo.

3.4.3.2. Gráficos e Impresión

3.4.3.2.1. El Graph Editing Framework (GEF)

GEF es el paquete de software que es el fundamento de los diagramas que aparecen en el Panel de Edición. GEF fué una parte integral de ArgoUML pero ha sido separada. Al igual que ArgoUML es un proyecto de código abierto disponible via Tigris [http://www.tigris.org].

3.4.3.2.2. Precision Graphics Markup Language (PGML)

PGML es el formato de almacenamiento actual para la información de diagrama usado en ArgoUML. En el futuro, PGML será sustituido por el formato UML 2.0 Diagram Interchange.

3.4.3.2.3. Aplicaciones Que Abren PGML

PGML is un predecesor de SVG (mira Sección 3.4.3.2.5, " Scalable Vector Graphics (SVG) ". Fué abandonado por el W3C Consortium.

Actualmente no hay otras herramientas que conozcamos trabajando en PGML.

3.4.3.2.4. Imprimiendo Diagramas

Seleciona un diagrama, luego vete a `Archivo#Exportar Diagramas`. Puedes generar formatos GIF, PostScript, Encapsulated PostScript o SVG.

3.4.3.2.5. Scalable Vector Graphics (SVG)

Un formato de gráficos vectoriales estandar del World Wide Web Consortium (W3C) (http://www.w3.org/TR/SVG/ [http://www.w3.org/TR/SVG/]).

Está soportado por los navegadores modernos, pero tambien puedes conseguir un plugin para navegadores antiguos en adobe.com [http://www.adobe.com].

3.4.3.2.6. Guardando Diagramas como SVG

1. Seleciona `.svg` como el tipo de archivo.
2. Teclea el nombre del archivo que quieras con la etiqueta `.svg` al final. Ejemplo `midiagramauml.svg`

Et viola! SVG! Pruebalo y juega con el zoom un poco... No son perfectos, así que si conoces algo sobre representar bonitos SVG haznoslo saber.

La mayoría de los navegadores modernos soportan SVG. Si el tuyo no lo hace, prueba Firefox [http://www.mozilla.com/firefox/] o consigue un plugin para tu navegador actual en adobe.com [http://www.adobe.com]

Nota

No tendras barras de desplazamiento para tu SVG a menos que este embebido en HTML. ¡Buena suerte y haznos saber que encuentras!

3.4.3.3. XMI

ArgoUML soporta archivos XMI 1.0, 1.1, y 1.2 que contengan modelos UML 1.3 y UML 1.4. Para una mejor compatibilidad con ArgoUML, exporta tus modelos usando UML 1.4 y XMI 1.1 o 1.2. Asegurate de desactivar cualquier extensión propietaria (tal como datos de diagrama de Poseidon).

Con versiones UML anteriores a UML 2.0, no es posible salvar información de diagrama, asi que no serán transferidos diagramas.

Existe tambien una herramienta que convierte XMI a HTML. Para mas información, mira

http://www.objectsbydesign.com/projects/xmi_to_html_2.html
[http://www.objectsbydesign.com/projects/xmi_to_html_2.html].

3.4.3.3.1. Usando XMI de Rational Rose

...

3.4.3.3.2. Usando Models Creados por Poseidon

En el diálogo `Exportar proyecto a XMI`, asegurate de dejar en blanco la selección de `Guardar con datos de diagrama`.

3.4.3.3.3. Usando Modelos Creados por MagicDraw

...

3.4.3.3.4. XMI Compatibilidad con otras versiones de ArgoUML

Las versiones de ArgoUML anteriores a 0.19.7 soportaban UML 1.3/XMI 1.0. Despues de este tiempo, el formato de almacenamiento es UML 1.4/XMI 1.2 el cual no es compatible hacia atras. Posteriores versiones de ArgoUML leerán proyectos escritos con versiones antiguas, pero no vice versa. Si puedes necesitar volver a una version antigua de ArgoUML deberías ser cuidadoso de guardar una copia de seguridad de tus viejos proyectos.

Adicionalmente, si escribes archivos XMI que necesitan ser leidos por otras herramientas, deberías tomar en cuenta las diferentes versiones. Las herramientas de modelado UML mas modernas deberían leer 1.4, pero puedes tener generadores de código integrados o otras herramientas que están atadas a UML 1.3.

3.4.3.3.5. Importando Otros Formatos XMI dentro de ArgoUML

La compatibilidad XMI entre herramientas de modelado UML ha mejorado con los años, pero puedes tener aún problemas ocasionalmente.

ArgoUML no leerá archivos XMI files que contengan modelos UML 1.5 o UML 2.0, pero debería ser capaz de abrir la mayoría de los archivos UML 1.4 y UML 1.3. Si encuentras uno que no puede abrir, por favor informa del bug para que un desarrollador pueda investigar.

3.4.3.3.6. Generando Formato XMI

Selecciona el comando `Exportar# Archivo como XMI` y escoge un nombre de archivo.

3.4.3.4. Generación de Código

3.4.3.4.1. Código Generado por ArgoUML

Es posible compilar tu código generado con ArgoUML, si bien aún necesitas implementar los cuerpos de metodos, para obtener resultados utiles.

3.4.3.4.2. Generando Código para Métodos

Hasta el momento no puedes escribir código para métodos (operaciones) dentro de ArgoUML. El panel de fuentes es editable, pero los cambios son ignorados. ArgoUML es una herramienta de diseño puro por ahora, no hay funcionalidad IDE pero el deseo está ahí. Puedes considerar usar Forte y ArgoUML juntos—¡Es un buen rodeo del problema!

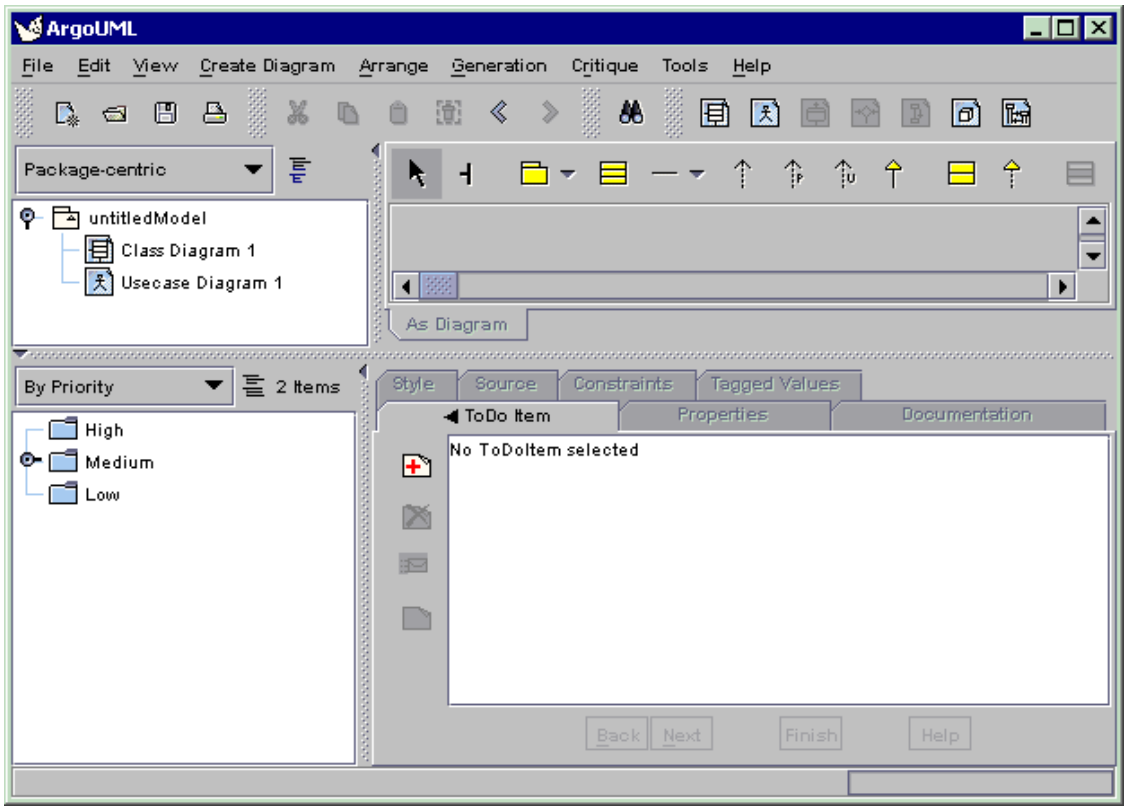
Puedes ayudarnos aquí si quisieras!

3.4.4. Trabajando Con Criticas de Diseño

3.4.4.1. Los Mensajes del Panel de Taréas Pendientes de las Criticas de diseño

¿Donde estamos ahora? Un nuevo proyecto ha sido creado, y está conectado al archivo `PrimerProyecto.argo`. Figura 3.10, “ Ventana de ArgoUML Habiendo Guardado `PrimerProyecto.zargo` ” muestra como tu ventana de ArgoUML debería aparecer en está etapa.

Figura 3.10. Ventana de ArgoUML Habiendo Guardado `PrimerProyecto.zargo`



El proyecto contiene un paquete de alto nivel llamado `untitledModel`, el cual contiene un diagrama de clases y un diagrama de casos de uso.

Si miramos cuidadosamente a la pantalla, podemos ver que la carpeta "Medium" en el panel de Taréas Pendientes (el panel de abajo a la izquierda) debe contener algunos elementos, ya que su icono de activación  está representado.

Hacer click en este icono abrirá la carpeta "Medium". Una carpeta abierta es indicada por el icono  .

Pero que es este Panel de “Taréas Pendientes” de todas formas. No has grabado nada aún que ha de ser hecho, así que donde se originan esos elementos.

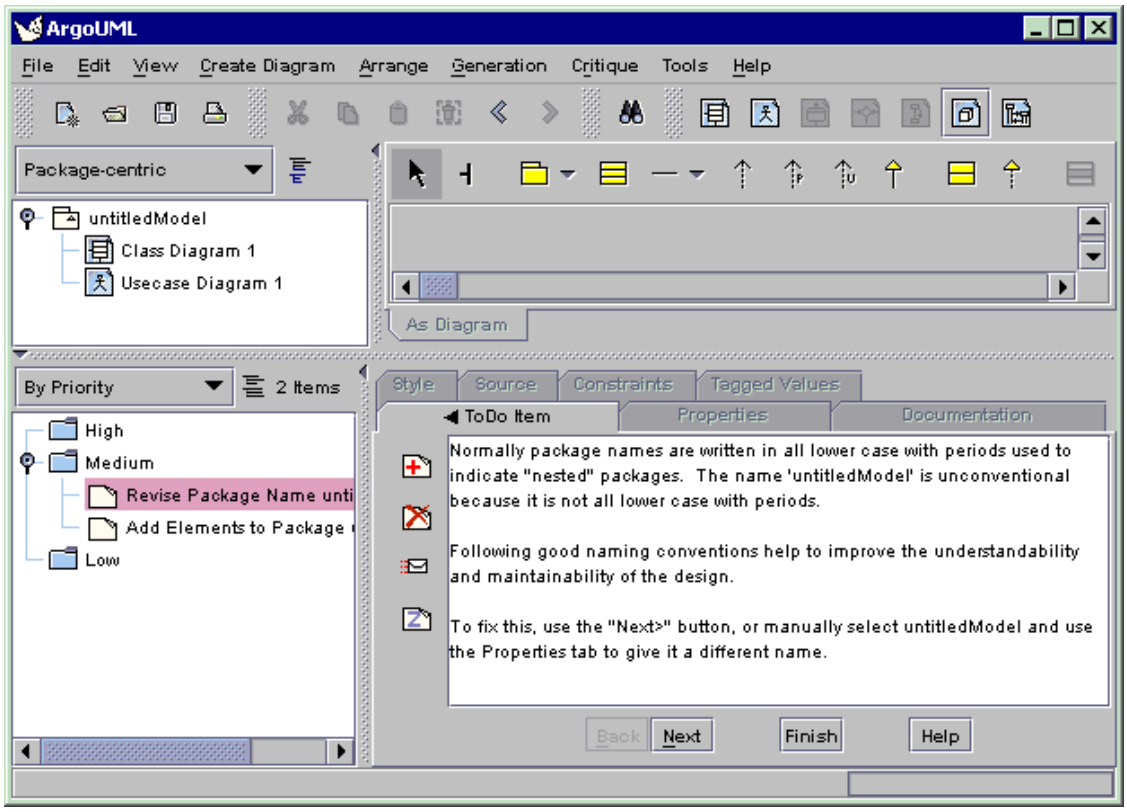
La respuesta el simple, y es al mismo tiempo uno de los puntos fuertes de ArgoUML. Mientras estas tra-

bajando en tu modelo UML, tu trabajo es monitorizado continuamente y de forma invisible por un trozo de código llamado una *crítica de diseño*. Esto es como un mentor personal que vigila por encima de tu hombro y te notifica cada vez que ve algo cuestionable en tu diseño.

Las criticas son bastante poco entrometidas. Te dan una advertencia amigable, pero no te fuerzan dentro de principios de diseño que no quieres o no te gusta seguir. Tomemos un vistazo de que nos están diciendo las críticas. Haz click en el icono  cerca de la carpeta `Medium`, y haz click en el elemento `Revisa el nombre del paquete UntitledModel` .

Figura 3.11, " Ventana ArgoUML Mostrando el Elemento de Crítica `Revisa el Nombre del paquete UntitledModel` " muestra como debería verse tu pantalla ahora.

Figura 3.11. Ventana ArgoUML Mostrando el Elemento de Crítica `Revisa el Nombre del paquete UntitledModel`



Observa que tu selección está destacada en rojo en el Panel de Taréas Pendientes, y que una explicación completa aparece ahora en el Panel de Detalles (el panel de abajo a la derecha). Puedes tener que redimensionar tu Panel de Detalles o desplazarloa hacia abajo para ver el mensaje completo como se muestra en nuestro ejemplo.

Lo que ArgoUML está intentando decirte es que normalmente, los nombres de paquetes están escritos en minusculas. El paquete principal por defecto creado por ArgoUML se llama `untitledModel` y por tanto viola un principio de diseño. (Realmente, este podría ser considerado como un bug dentro de ArgoUML, pero es adecuado para demostrar el funcionamiento de las críticas).

En este punto, puedes escoger cambiar el nombre del paquete manualmente, para imponer silencio en la crítica de diseño por algún tiempo o permanentemente, o para requerir una explicación mas extensa por

e-mail de un experto.

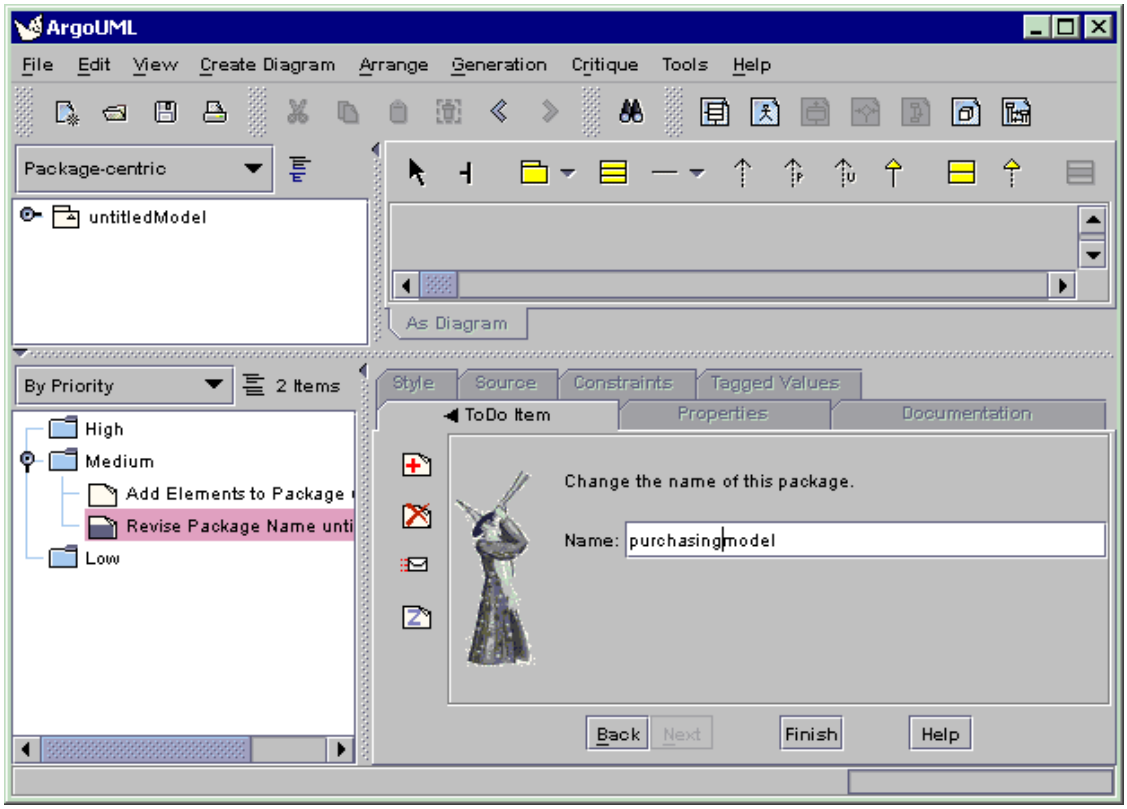
No haremos nada de esto (volveremos a ello cuando hablemos sobre las críticas de diseño con mas detalle) pero usaremos otra práctica caracteristica de ArgoUML—una caracteristica de autocorrección.

Para hacerlo, solo hac click en el boton `Siguiente (Próximo)` del Panel de Detalles. Esto causara que un aistente de renombar sea mostrado dentro del panel de propiedades, proponiendo usar el nombre `untitledmodel` (todo en minusculas).

3.4.4.2. Críticas de Diseño Funcionando: El Asistente de Renombrar Paquete

Sustitulye el nombre `untitledmodel` con `purchasingmodel`, y haz click el el botton `Terminar`. Figura 3.12, " Ventana de ArgoUML Mostrando el Asistente de Crítica para Renombrar el Paquete " muestra como la ventana de ArgoUML aparecerá ahora.

Figura 3.12. Ventana de ArgoUML Mostrando el Asistente de Crítica para Renombrar el Paquete



Observa ahora como la nota de la crítica de diseño en el panel de Taréas Pendientes desaparece, dejando solo la nota `Añade elementos al paquete purchasingmodel` en la lista de Taréas Pendientes.

Si esto no ocurre al momento, espera algunos segundos. ArgoUML hace uso intensivo de muchos hilos de ejecución que se ejecutan en paralelo. Esto puede causar demoras de algunos segundos antes de que la información se actualize en la pantalla.

El cambio del nombre del paquete debería ser reflejado en el explorador, en la esquina superior izquierda de tu ventana de ArgoUML.

Ahora estamos preparados para crear nuestro primer diagrama UML, un diagrama de Casos de Uso, pero primero guardemos lo que hemos hecho hasta ahora.

Haz click en el elemento de menú `Archivo`, y seleciona `Guardar Proyecto`. Ahora puedes salir de forma segura de ArgoUML sin perder tu trabajo hasta el momento, o seguir creando tu primer diagrama.

3.5. El Casos de Estudio (A escribir)

A escribir...

...

Capítulo 4. Captura de Requerimientos

4.1. Introducción

La captura de requerimientos el el proceso de identificar que quiere el “cliente” del sistema propuesto.

La clave en esta etapa es que estamos en el dominio de problema. En esta etapa debemos describir todo desde la perspectiva del “cliente” y en el lenguaje del “customer”.

El mayor riesgo que tenemos en la caputra de requerimientos es empezar pensando en terminos de posibles soluciones. Eso debe esperar hasta la *Fase de Analisis* (mira Capítulo 5,). Uno de los pasos de la Fase de Analisis será tomar los resultados de la Fase de Requerimientos y refundirlos en el lenguaje de la solución estimada.

Recuerda que estamos usando un proceso *incremental* y *iterativo*.

Podemos tranquilamente volver atras otra vez al proceso de requerimientos mientras desmontamos el problema en trozos mas pequeños, cada uno de los cuales debe tener sus requerimientos capturados.

Ciertamente volveremos atras a traves de la fase de requerimientos en cada iteración mientras buscamos definir los requerimientos del sistema mas y mas.

Nota

La unica parte de la notación de los requerimientos especificada por el estandar UML es el diagrama de casos de uso. El resto es especifico del proceso. El proceso descrito en este capitulo está muy inspirado en el Rational Unified Process.

4.2. El Prodeso de Captura de Requerimientos

Empezamos con una visión general del problema que estamos resolviendo y las areas clave de funcionalidad que debemos tratar en cualquier solución. Este es nuestro *documento de visión*, y debería ser de solo algunas páginas de longitud.

Por ejemplo la visión general de un cajero automatico (automated teller machine; ATM) puede ser que debería soportar lo siguiente.

1. Depositos monetarios, reembolsos monetarios y consultas de cuenta por los clientes.
2. Mantenimiento del equipamiento por los ingenieros del banco, y descarga de depositos y carga de dinero por la sucursal local del banco.
3. Auditorias de todas las actividades enviadas al sistema central del banco.

Desde esta visión general podemos extraer las actividades principales del sistema, y los agentes externos (personas, equipamiento) que están involucradas en estas actividades. Estas actividades son conocidad como *casos de uso* y los agentes externos son conocidos como *actores*.

Los actores pueden ser personas o maquinas. Desde un punto de vista practico es beneficioso conocer la parte implicada detras de cada maquina, puesto que solo ellos serán capaces de tratar con el proceso de captura de requerimientos.

Los casos de uso deberías ser actividades significativas para el sistema. Por ejemplo el uso por el cliente del Cajero Automatico es un caso de uso. Introducir un numero PIN no lo es.

Hay una zona intermedia entre estos dos extremos. Como veremos a menudo es util dividir casos de uso muy grandes en pequeños subcasos de uso. Por ejemplo podemos tener subcasos de uso cubriendo depositos de dinero, retiradas de dinero y consultas de cuenta.

No hay una regla clara y rápida. Algunos arquitectos preferirán un pequeño numero de casos de uso relativamente grandes, otros en en cambio preferirán un grán número de pequeños casos de uso. Una regla util a primera vista es que cualquier proyecto practico debería requerir no mas de alrededor de 30 casos de uso (si necesita mas, debería ser dividido en proyectos separados).

Luego mostramos la relación entre casos de uso y actores en uno o mas diagramas de casos de uso. Para un proyecto grande puede ser necesario mas de un diagrama. Normalmente los grupos de casos de uso relacionados son mostrados en un mismo diagrama.

Debemos luego dar una especificación mas detallada de cada caso de uso. Esto cubre su comportamiento normal, comportamientos alternativos y cualquier precondición y postcondición. Esto se refleja en un documento conocido como *especificación de caso de uso* o *escenario de caso de uso*.

Finalmente, puesto que los casos de uso son funcionales en su naturaleza, necesitamos un documento para capturar los requerimientos no funcionales (capacidad, rendimiento, necesidades de entorno, etc). Estos requerimientos son capturados en un documento conocido como una *especificación suplementaria de requerimientos*.

4.2.1. Pasos del Proceso

Los pasos en el proceso de captura de requerimientos pueden ser resumidos como sigue.

1. Captura una vista general del problema, y las caracteristicas deseadas de su solución en el *documento de visión*.

2. Identificar los *casos de uso* y *actores* desde el documento de visión y mostrar sus relaciones en uno o mas *diagramas de casos de uso*.

3. Da *especificaciones de casos de uso* detalladas para cada caso de uso, cubriendo el comportamiento normal y alternativo, precondiciones y postcondiciones.

4. Captura todos los requerimientos no funcionales en una *especificación de requerimientos suplementarios*.

En cualquier proceso de desarrollo iterativo, priorizaremos, y las iteraciones tempranas se enfocaran en capturar el comportamiento clave del los casos de uso mas importantes.

La mayoría de los procesos de captura de requerimientos modernos están deacuerdo con que es esencial que un representante autorizado del cliente esté completamente involucrado a traves del proceso.

4.3. Salida del Proceso de Captura de Requerimientos

Casi todo el resultado del proceso de captura de requerimientos es documental. El unico diagrama es el diagrama de casos de uso, mostrando las relaciones entre casos de uso y actores.

4.3.1. Documento de Visión

Las secciones tipicas de este documento serían como sigue.

- *Resumen*. Una declaración del contexto, problema y objetivo de la solutión.
- *Objetivos*. Que estamos intentando alcanzar (y como deseamos alcanzarlo).
- *Contexto de Mercado* o *Convenios Contractuales*. Para un desarrollo guiado por el mercado, esto debería indicar mercados objetivo, diferencidores competitivos, eventos motivadores y cosas así. Para un desarrollo contractual esto debería explicar los factores clave contractuales.
- *Partes Implicadas*. Los usuarios (en el sentido más amplio) del sistema. Muchos de ellos se mapearán en actores, o equipamiento de control que se mapea en actores.
- *Caracteristicas clave*. Los aspectos funcionales clave de la solución deseada al problema al nivel mas alto. Estos se mapearán ampliamente en los casos de uso. Es de ayuda poner algo de priorización aquí.
- *Limitaciónes*. Una visión general de los parametros no funcionales del sistema. Estos serán tratados detalle en la especificación de requerimientos suplementarios.
- *Apendice*. Un listado de los actores y casos de uso que serán necesarios para cumplir esta visión. Es util enlazar a estos desde las secciones tempranas para asegurar una cobertura exahustiva.

4.3.2. Diagrama de Casos de Uso

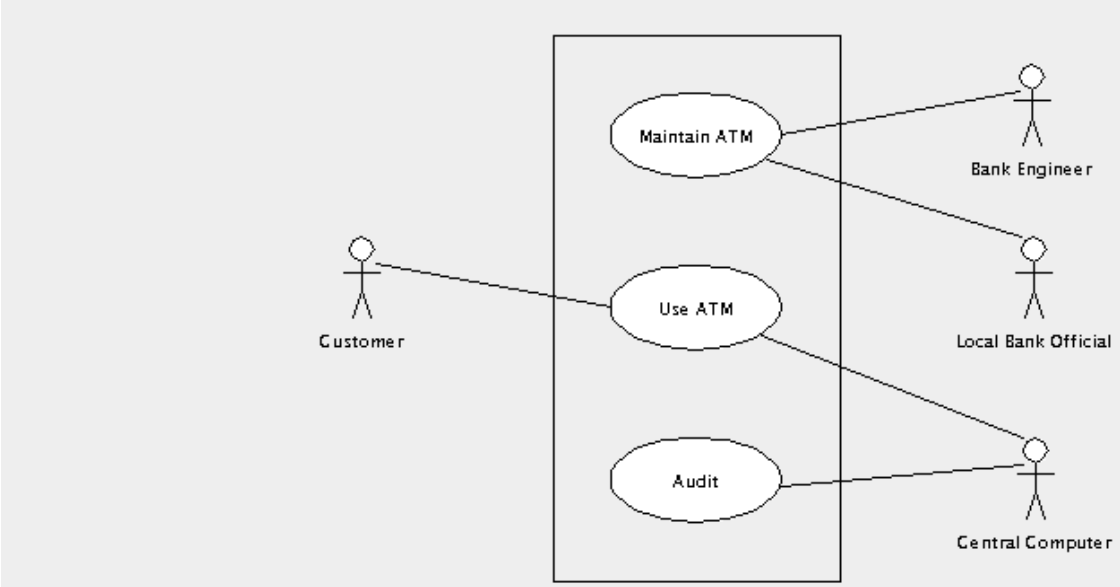
El documento de visión ha identificado los casos de uso y los actores. El diagrama de casos de uso captura como interaccionan. En nuestro ejemplo de Cajero Automatico hemos identificado “cliente usa cajero”, “mantener cajero” y “auditar” como los tres casos de uso principales. Hemos identificado “cliente”, ingeniero de mantenimiento“, ” “oficial de sucursal” y “sistema central” como los actores.

Figura 4.1, “ Diagrama de casos de uso basico para un sistema de Cajero Automatico ” muestra como esto puede ser visualizado en un diagrama de casos de uso. Los casos de uso son mostrados como ovalos, los actores como monigotes (incluso cuando son maquinas), con lineas (conocidas como *asociaciones*) conectando los casos de uso a los actores que están involucrados en ellos. Un cuadro alrededor de los casos de uso enfatiza la frontera entre el sistema (definido por los casos de uso) y los actores que son externos.

Nota

No todos los analisis gustan de usar un cuadro alrededor de los casos de uso. Es un asunto de opción personal.

Figura 4.1. Diagrama de casos de uso basico para un sistema de Cajero Automatico



Las siguientes secciones muestran como el diagrama de casos de uso básico puede ser extendido para mostrar información adicional sobre el sistema que está siendo diseñado.

4.3.2.1. Actores Activos y Pasivos

Los actores *Activos* inician la interacción con el sistema. Esto puede ser mostrado colocando una flecha en la asociación desde el actor apuntando hacia el caso de uso. En le ejemplo del Cajero Automatico, el cliente es un actor activo.

La interacción con los actores *pasivos* es iniciada por el sistema. Esto puede ser mostrado colocando una flecha en la asociación desde el caso de uso apuntando hacia el actor. En el ejemplo del Cajero Automatico el sistema central es un actor pasivo.

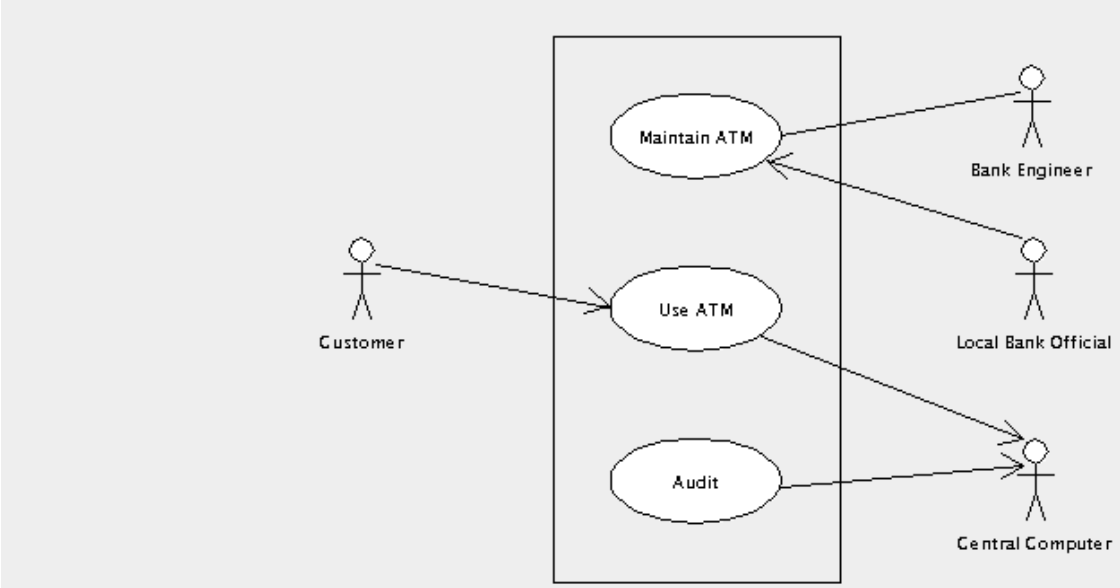
Este es un buen ejemplo donde la flecha ayuda, puesto que nos permite distinguir un sistema conducido por eventos (el Cajero Automático inicia la interacción con el sistema central) de un sistema de consulta continua (el sistema central interroga al Cajero Automático de tiempo en tiempo).

Donde un actor puede ser, una de dos, activo o pasivo, dependiendo de las circunstancias, la flecha puede ser omitida. En el ejemplo del Cajero Automático el ingeniero del banco se ajusta a esta categoría. Normalmente el está activo, poniendose en funcionamiento en cicles regulares para dar servicio al Cajero. Sin embargo is el Cajero Automático detecta un fallo, puede convocar al ingeniero para repararlo.

El uso de flechas en asociaciones es referido como la *navegación* de la asociación. Veremos esto usado en otro lugar en UML mas tarde.

Figura 4.2, “ Diagrama de casos de uso para un Cajero Automático mostrando navegación. ” muestra el diagrama de casos de uso del Cajero Automático con navegación representada.

Figura 4.2. Diagrama de casos de uso para un Cajero Automático mostrando navegación.



4.3.2.2. Multiplicidad

Puede ser util mostrar la *multiplicidad* de asociaciones entre actores y casos de uso. Con esto queremos decir cuantas instancias de un actor interacciona con cuantas instancias del caso de uso.

Po defecto asumimos que una instancia de un actor interacciona con una instancia de un caso de uso. En otros casos podemos etiquetar la multiplicidad de un extremo de la asociación, una de dos con un número para indicar cuantas instancias están involucradas, o con un rango separado por dos puntos (`..`). Un asterisco (`*`) es usado para indicar un número arbitrario.

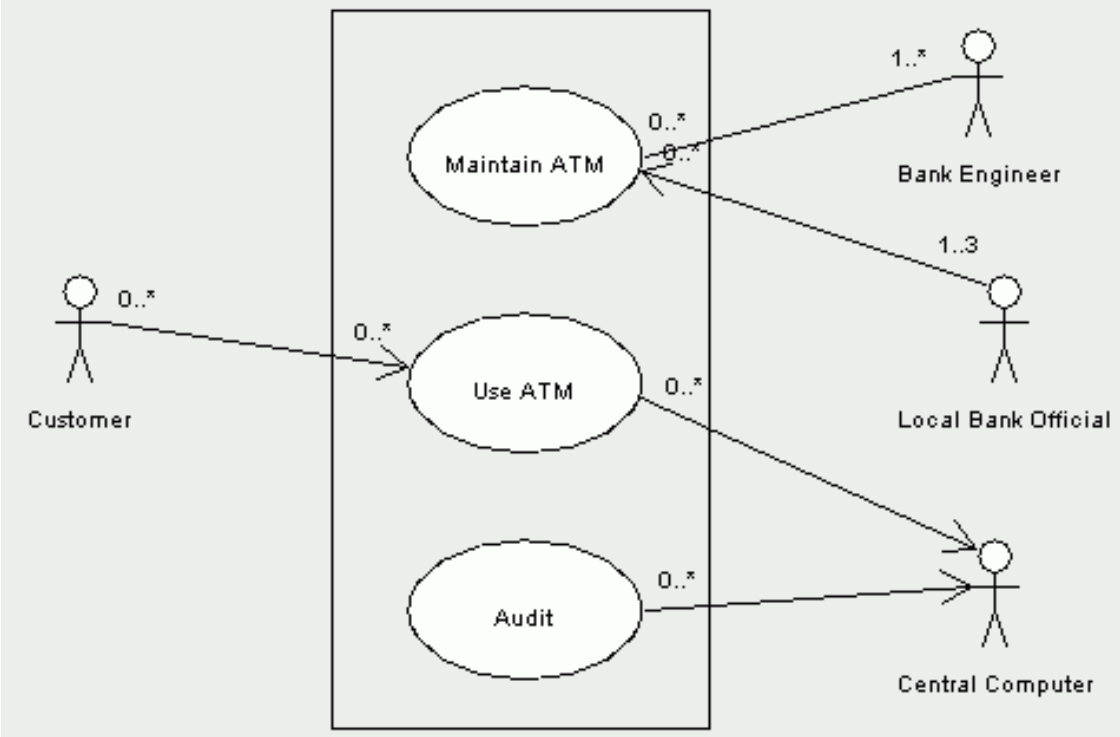
En el ejemplo del Cajero Automático, solo hay un sistema central, pero el puede estar gestionando cualquier número de usos de Cajero Automático. Por lo tanto colocamos la etiqueta `0..*` en el extremos del caso de uso. No se necesita etiqueta en el otro extremo, ya que por defecto es uno.

Un banco local tendrá hasta tres oficiales autorizados para cargar y descargar los Cajeros Automáticos. Asi en el extremo del actor de la relación con el caso de uso `Mantenimiento Cajero Automático`, colocamos la etiqueta `1..3`. Ellos pueden estar tratando con cualquier número de Cajeros Automaticos, así en el otro extremo colocamos la etiqueta `0..*`.

Puede haber cualquier número de clientes y puede haber cualquier número de sistemas de Cajero Automático que pueden usar. Así a cada extremo de la asociación colocamos la etiqueta `0..*`.

Figura 4.3, “ Diagrama de casos de uso para un sistema de Cajero Automático mostrando multiplicidad. ” muestra el diagrama de casos de uso del Cajero Automático con multiplicidad representada.

Figura 4.3. Diagrama de casos de uso para un sistema de Cajero Automático mostrando multiplicidad.



La multiplicidad puede abarrotar un diagrama, y a menudo no se muestra, excepto donde es crítico comprenderlo. En el ejemplo de Cajero Automatico solo elegiriamos mostrar `1..3` contra el oficial del banco, ya que todos los demas son obvios por el contexto.

4.3.2.3. Jerarquías de Casos de Uso

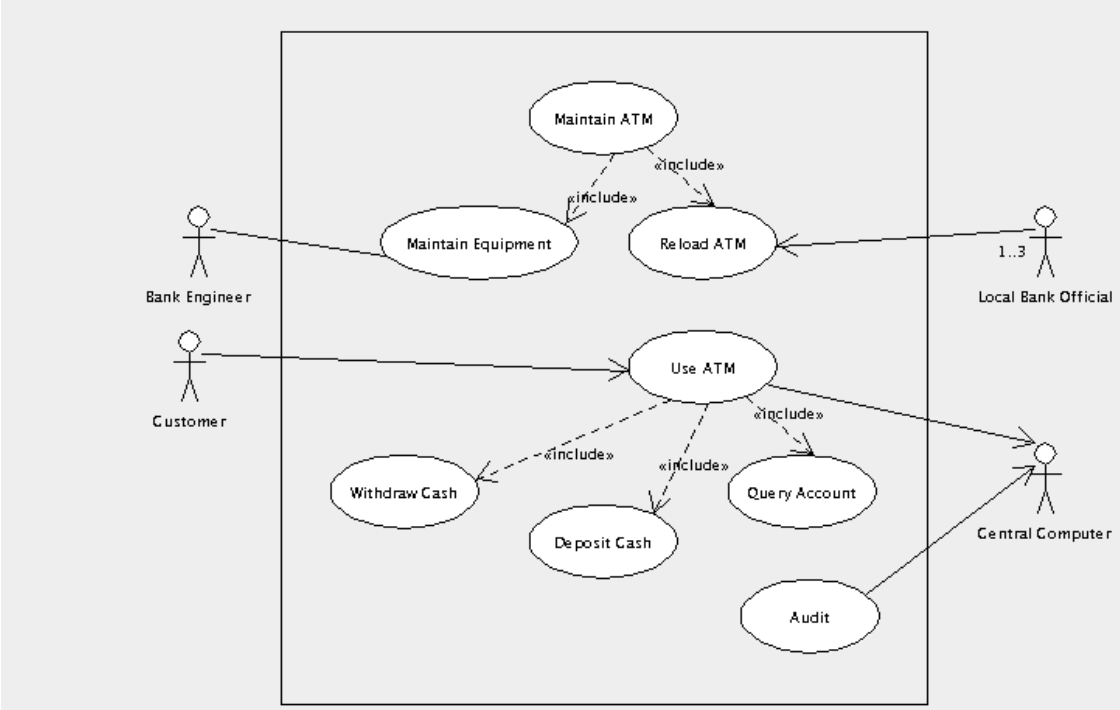
En nuestro ejemplo de Cajero Automatico hasta ahora tememos solo tres casos de uso para describir el comportamiento del sistema. Mientras los casos de uso siempre deberían describir un trozo significativo del comportamiento del sistema, si son demasiado generales pueden ser dificiles de describir.

Podríamos por ejemplo definir el comportamiento del caso de uso “Uso Cajero” en terminos del comportamiento de los tres casos de uso mas simples, “Depositar Dinero”, “Retirar Dinero” y “Consultas de Cuenta”. El caso de uso principal podría ser especificado *incluyendo* el comportamiento de los casos de uso subsidiarios necesarios.

Similarmente el caso de uso “Mantener Cajero” podría ser definido en terminos de dos casos de uso “Mantener Equipamiento” y “Recargar Cajero”. En este caso los dos actores involucrados en el caso de uso principal están realmente solo involucrados en uno u otro de los dos casos de uso subsidiarios y esto puede ser mostrado en el diagrama.

La descomposición de un caso de uso en subcasos de uso mas simples es mostrada en UML usando una *relación de inclusión*, una flecha punteada desde el caso de uso principal hasta el subsidiario, con la etiqueta *«include»*.

Figura 4.4. Diagrama de caso de uso para un sistema de Cajero Automático mostrando relaciones de inclusión.



Las relaciones de inclusión son buenas para desmantelar los comportamientos de casos de uso en jerarquias. Sin embargo podemos tambien querer mostrar un caso de uso que es una *extensión* de un caso de uso existente para atender a una circunstancia particular.

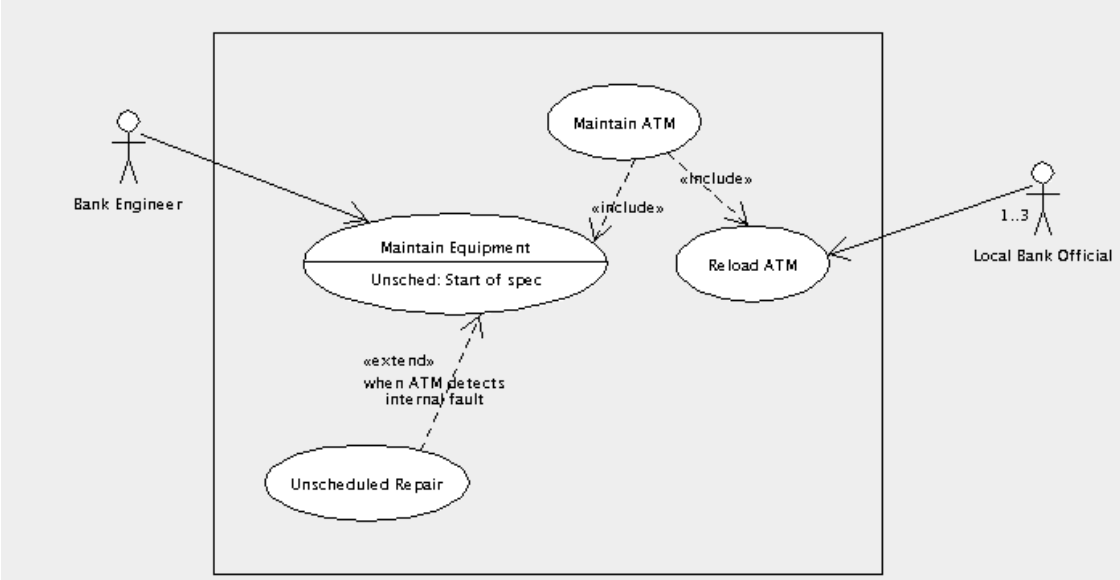
En el ejemplo del Cajero tenemos un caso de uso cubriendo la rutina de mantenimiento del Cajero, “Mantener Equipamiento”. Tambien queremos cubrir el caso especial de una reparación no programada causada por la detección de un fallo interno por parte del Cajero.

Esto está mostrado en UML por la relación de *extensión*. En el caso de uso principal, especificamos un nombre para un lugar en la descripción, donde una extensión del comportamiento podría ser adjuntada. El nombre y lugar son mostrados en un compartimento separado dentro del ovalo del caso de uso. La representación de relación de extensión es la misma que la de la relación de inclusión, pero con la etiqueta *«extend»*. Paralelamente a la relación de extensiñon, especificamos la condicion bajo la cual ese comportamiento será adjuntado.

Figura 4.5, “ Diagrama de casos de uso para un sistema de Cajero Automático mostrando una relación de extensión. ” muestra el diagrama de casos de uso del Cajero con una relación de extensión a un caso de uso para reparaciones no programadas. El diagrama se está volviendo bastante complejo, y así lo hemos dividido en dos, uno para el lado del mantenimiento, el otro para el uso del cliente y auditoría.

El caso de uso “Mantener Equipamiento” define un nombre “Unsched”, al comienzo de su descripción. El caso de uso extendido “Unscheduled Repair” es adjuntado ahí cuando el Cajero detecta un error interno.

Figura 4.5. Diagrama de casos de uso para un sistema de Cajero Automático mostrando una relación de extensión.



Los casos de uso pueden ser enlazados juntos de una forma adicional. Un caso de uso puede ser una *generalización* de un caso de uso subsidiario (o alternativamente el subsidiario es una *especializatión* del caso de uso principal).

Esto es muy parecido a la relación de extension, pero sin la limitación de puntos de extensión especificos en los cuales el caso de uso principal puede ser extendido, y sin condiciones sobre cuando puede ser usado el caso de uso subsidiario.

La generalización está representada en un diagrama de casos de uso por una flecha con una linea continua y una punta de flecha blanca desde el subsidiario al caso de uso principal.

Esto puede ser util cuando un caso de uso subsidiario especializa el comportamiento del caso de uso principal en un gran número de posiciones y bajo un amplio rango de circunstancias.

Sin embargo la falta de alguna restricción hace la generalización muy dificil de especificar con precisión. En general usa una relación de extensión en su lugar.

4.3.3. La Especificación de Casos de Uso

Cada caso de uso debe ser documentado para explicar en detalle el comportamiento que está especificando. Este documento es conocido por diferentes nombres en diferentes procesos: *especificación de caso de uso*, *escenario de caso de uso* o incluso (confusamente) solo *caso de uso*.

Un caso de uso tipico incluirá las siguientes secciones.

- *Nombre*. El nombre del caso de uso al que esto se refiere.
- *Objetivo*. Un resumen de una o dos lineas de que realiza este caso de uso *por sus actores*.
- *Actores*. Los actores involucrados en este caso de uso, y cualquier contexto con respecto a su participación.

Nota

Esto no debería ser una descripción del actor. Eso debería estar asociado con el actor en el diagrama de casos de uso.

- *Pre-condición*. Sería mejor llamarlas “pre-asumciones”, pero el termino usado en todos sitios es pre-condiciones. Es una declaración de cualesquiera asumciones de simplificación que podemos hacer al comienzo del caso de uso.

  En el ejempli del Cajero Automatico, podemos hacer la asumción para el caso de uso de “Mantener Equipamiento” que un ingeniero está siempre disponible, y no necesitamos preocuparnos sobre el caso de que una visita de mantenimiento de rutina se halla dejado pasar.

  ## Atención

  Evita pre-condiciones todo lo posible. Necesitas tener la absoluta certeza de que las pre-condiciones caben bajo todas las posibles circunstancias. Si no tu sistema estará debilmente especificado y por lo tanto fallara cuando la pre-condición no es cierta. Alternativamente, cuando no tienes la certeza de que la pre-condición es siempre cierta, necesitaras especificar un segundo caso de uso para manejar la pre-condición siendo falsa. En el primer caso, las pre-condiciones son una fuente de problemas, en el segundo una fuente de mas trabajo.

- *Flujo Básico*. La secuencia lineal de pasos que descruben el comportamiento de el caso de uso en el escenarío “normal”. Donde un caso de uso tiene un número de escenarios que podrían ser normales, uno es seleccionado arbitrariamente. Especificar el flujo básico está descrito con más detalle mas Sección 4.3.3.1, “” abajo.

- *Flujos Alternativos*. Unas series de secuencias lineales describiendo cada uno de los comportamientos alternativos al flujo básico. Especificar flujos alternativos está descrito con mas detalle en Sección 4.3.3.2, “”.

- *Post-condiciones*. Sería mejor llamarlas “post-asumciones”. Esta es una especificación de cualesquiera asumciones que podemos hacer al final del caso de uso. Son mas utiles donde el caso de uso es uno de una serie de casos de uso subsidiarios que estan incluidos en un caso de uso principal, donde pueden formar las pre-condiciones del siguiente caso de uso que va a ser incluido.

  ## Atención

  Como las pre-condiciones, las post-condiciones son mejor evitarlas. Colocan una carga en la especificación de los flujos de caso de uso, para asegurar que la post-condición siempre se mantiene. Por lo tanto son tambien una fuente de problemas y trabajo extra.

- *Requerimientos*. En un mundo ideal el documento de visión, los diagramas de casos de uso, las especificaciones de casos de uso y la especificacion de requerimientos suplementarios formarián los requerimientos para un proyecto.

  Para la mayoría de los desarrollos lideres del mercado, donde la propiedad de los requerimientos esta dentro del mismo negocio que el equipo que hará el desarrollo, este no es normalmente el caso. El departamento de marketing puede aprender captura de requerimientos basada en casos de uso y ana-

lisis para enlazar con las actividades primarias de sus consumidores.

Sin embargo para desarrollos externos por contrato, los consumidores pueden insistir en una “lista de caracteristicas” tradicional como la base del contrato. Cuando este es el caso, esta sección de la especificación de casos de uso debería enlazar con las caracteristicas del contrato que son cubiertas por el caso de uso.

Esto es hecho a menudo a traves de una herramienta de terceros que puede enlazar documentos, proporcionando una prueba automatizada de cobertura de requisitos, en tal caso esta sección no es necesaria, o puede ser generada automaticamente.

El tamaño final de la especificacíon de caos de uso dependerá de la complejidad del caso de uso. Como pequeña regla, la mayoría de los casos de uso toman alrededor de 10-15 paginas para especificar, la mayoría de las cuales son flujos alternativos. Si el tuyo es mucho mas largo que esto, considera simplificar el caso de uso. Si el tuyo es mucho mas pequeño considera que el caso de uso esta describiendo una parte demasiado pequeña del comportamiento.

4.3.3.1.

Todos los flujos en una especificación de casos de uso son lineales (esto es que no hay ramas condicionales). Cualesquiera elecciones en los flujos son manejadas especificando otro flujo alternativo que recoge el punto de elección. Es importante recordar que estamos especificando comportamiento aqui, no programandolo.

Un flujo es especificado como una serie de pasos numerados. Cada paso debe implicar alguna interacción con un actor, o al menos generar un cambio que sea observable externamente por un actor. La captura de requerimientos no debería estar especificando comportamiento interno y oculto del sistema.

Por ejemplo nosotros podemos dar la siquiente secuencia de pasos para el flujo basico del casos de uso "Retirar Dinero" en nuestro ejemplo de Cajero Automatico.

1. Consumidor indica que se requiere recibo.
2. Consumidor introduce cantidad de dinero requerido.
3. Cajero Automatico verifica con el ordenador central que el consumidor puede realizar esta operación.
4. Cajero Automatico entrega el dinero a el consumidor.
5. Cajero Automatico emite recibo al consumidor.

Recuerda que esta es un sub-caso de uso incluido en el caso de uso principal “Usar Cajero Automatico”, el cual presumiblemente manejará la verificación de tarjetas y PINs antes de invocar este caso de uso incluido.

Nota

El primer paso no es una condición. Tomamos como nuestro flujo básico el caso donde el consumidor quiere un recibo. El caso donde de consumidor no quiere un recibo será un flujo alternativo.

4.3.3.2.

Esto captura los escenarios alternativos, como flujos lineales, mediante referencia la flujo básico. Inicialmente unicamente construimos una lista de los flujos alternativos.

A.
A.1. Consumidor no requiere recibo.

A.2. La cuenta del consumidor no soportará el retiro de dinero.

A.3. Comunicación con el ordenador central está interrumpida.

A.4. El consumidor cancela la transacción.

A.5. El consumidor falla a coger el dinero entregado.

A.
A.1.
A.1.
A.1.
1.
2.

4.3.3.3.

4.3.4.

Nota

-
-
-
-
-
-

4.4.

4.4.1.

4.4.2.

1.
2.

4.4.2.1.

1.

Nota

2.

1.
2.

4.4.3.

4.4.3.1.

1.
2.

Nota

4.4.3.2.

1.
2.

4.4.4.

4.4.4.1.

4.4.4.2.

Nota

4.4.4.3.

4.4.5.

Aviso

4.4.6.

Aviso

4.4.7.

Nota

4.5.

4.5.1.

4.5.1.1.

4.5.1.2.

4.5.1.3.

4.5.1.4.

- .
- .
- .
- .
- .

4.5.1.5.

4.5.1.6.

4.5.1.7.

- .
- .
- .
- .

- .
- .
- .

4.5.2.

4.5.3.

4.5.4.

4.5.5.

4.5.6.

Capítulo 5.

5.1.

5.1.1.

5.1.2.

5.1.3.

5.1.4.

5.1.5.

5.1.6.

5.2.

5.2.1.

5.2.2.

5.2.2.1.

5.3.

5.3.1.

5.3.1.1.

Aviso

5.3.2.

5.3.2.1.

5.3.3.

5.3.3.1.

5.3.3.2.

5.3.3.3.

5.3.4.

5.3.4.1.

5.3.4.2.

5.4.

5.4.1.

5.4.2.

5.4.3.

5.5.

5.5.1.

5.5.1.1.

5.5.2.

5.5.3.

5.6.

5.6.1.

5.6.2.

5.6.2.1.

5.7.

5.7.1.

5.7.1.1.

5.7.2.

5.7.2.1.

5.7.3.

5.7.4.

5.7.5.

5.7.5.1.

5.8.

5.9.

5.10.

5.10.1.

5.10.2.

5.10.2.1.

5.10.2.2.

5.10.3.

5.10.3.1.

5.10.4.

5.10.5.

Capítulo 6.

6.1.

...

6.1.1.

6.1.2.

6.1.3.

6.1.4.

6.1.5.

6.1.6.

6.1.7.

6.2.

6.2.1.

6.2.2.

6.2.2.1.

6.2.2.2.

6.2.2.3.

6.3.

6.3.1.

6.3.1.1.

6.3.2.

6.3.2.1.

6.3.2.2.

6.3.2.3.

6.3.3.

6.3.3.1.

6.3.3.2.

6.4.

6.4.1.

6.4.1.1.

6.4.1.2.

6.4.2.

6.4.2.1.

6.5.

6.5.1.

6.5.2.

6.5.2.1.

6.5.2.2.

6.5.3.

6.5.3.1.

6.5.3.1.1.

Figura 6.1.

Figura 6.2.

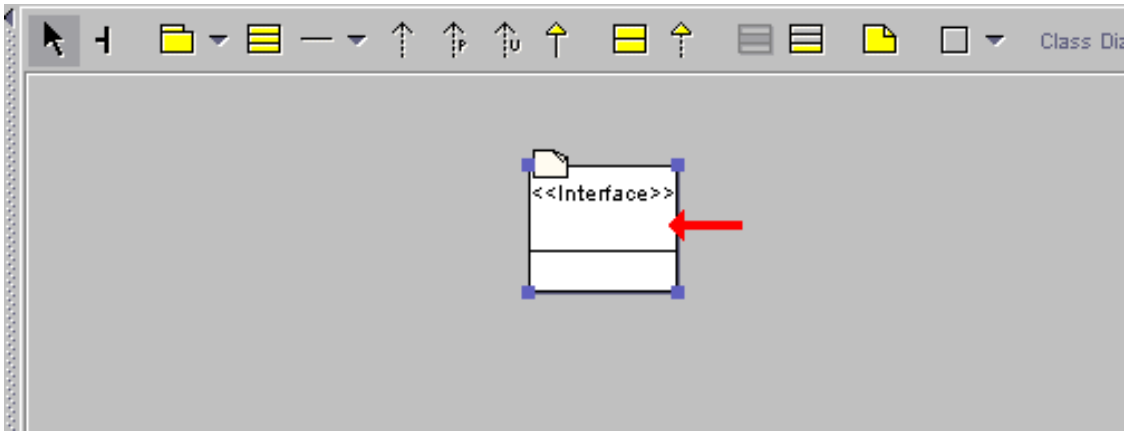


Figura 6.3.

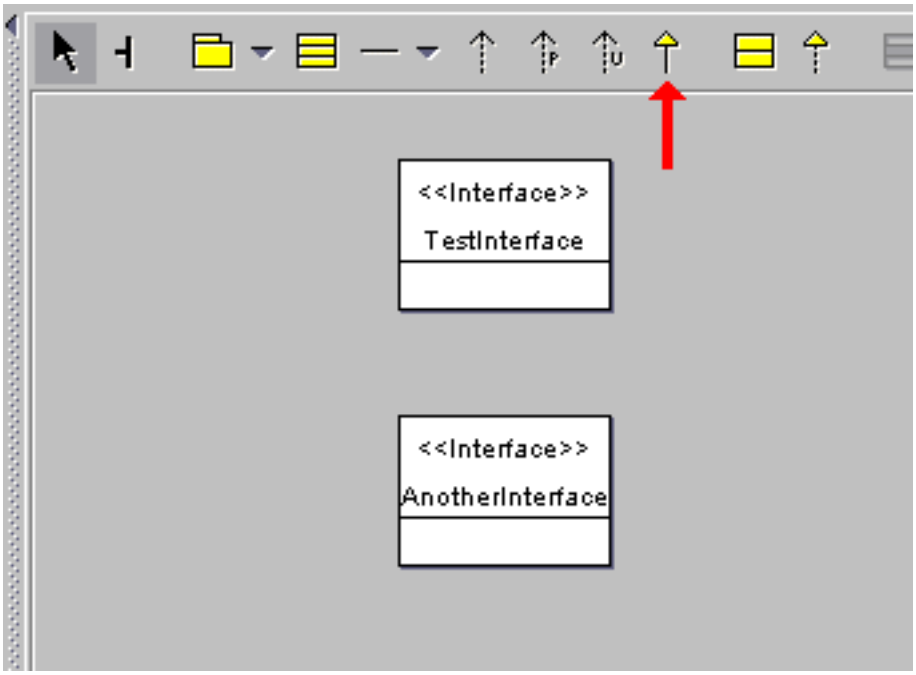
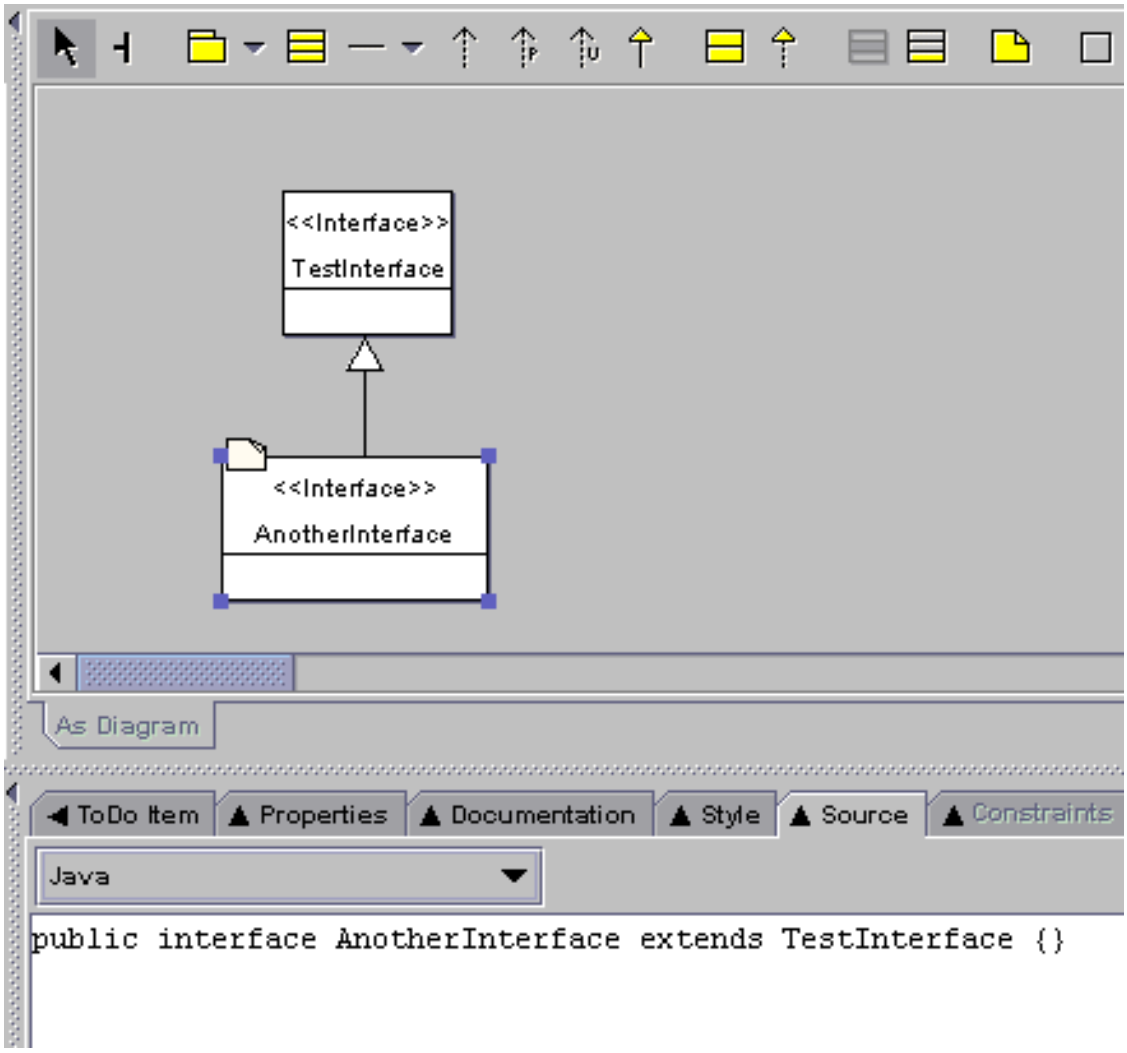


Figura 6.4.



6.5.3.2.

6.6.

Nota

6.6.1.

6.6.2.

6.6.2.1.

6.6.2.2.

6.6.3.

6.7.

6.7.1.

6.7.2.

6.7.2.1.

6.7.3.

6.8.

6.8.1.

6.8.2.

6.8.2.1.

6.8.2.2.

6.8.2.2.1.

6.8.2.2.2.

6.8.2.2.3.

6.8.2.3.

6.8.2.3.1.

6.8.2.3.2.

6.8.2.4.

6.8.2.5.

6.9.

6.9.1.

6.9.2.

6.9.3.

6.9.4.

6.9.5.

6.9.5.1.

6.9.5.1.1.

6.9.5.1.2.

6.9.5.1.3.

6.9.5.2.

6.9.5.2.1.

6.9.5.2.2.

6.9.5.3.

6.9.5.4.

6.10.

6.10.1.

6.10.1.1.

6.11.

6.11.1.

6.11.1.1.

6.11.2.

6.12.

6.12.1.

6.13.

6.13.1.

6.13.1.1.

6.13.2.

6.13.2.1.

6.13.3.

6.13.3.1.

6.13.3.2.

6.13.3.3.

6.14.

6.15.

6.15.1.

6.15.2.

6.15.2.1.

6.15.2.2.

6.15.3.

6.15.3.1.

6.15.3.2.

6.15.3.3.

6.15.4.

6.15.4.1.

6.15.5.

6.15.5.1.

6.15.6.

6.15.7.

6.15.8.

6.15.9.

Capítulo 7.

7.1.

7.2.

7.2.1.

7.2.2.

7.3.

7.3.1.

7.3.2.

7.4.

7.5.

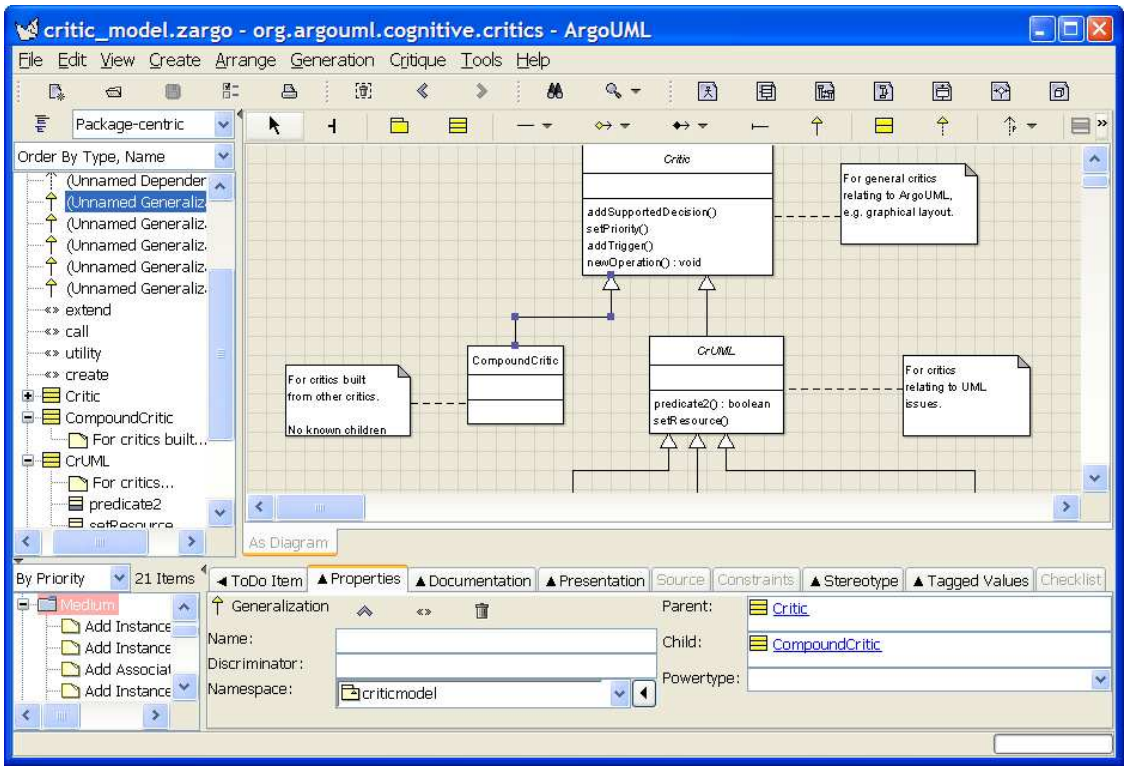
Parte 2.

Capítulo 8.

8.1.

.
.
.
.

Figura 8.1.



8.2.

8.2.1.

8.2.2.

8.2.2.1.

.
.
.

8.2.2.2.

8.2.2.3.

8.2.2.4.

8.2.3.

8.2.3.1.

8.2.4.

8.2.4.1.

8.2.5.

8.2.5.1.

Atención

8.2.5.2.

8.2.6.

8.2.7.

8.2.8.

8.2.9.

8.2.10.

8.3.

8.3.1.

8.4.

Capítulo 9.

9.1.

.
.
.
.
.

9.2.

.
.
.
.

9.3.

.
.
    .
    .
    .
    .
    .

9.4.

.
.
.
.
.
.
.

Capítulo 10.

10.1.

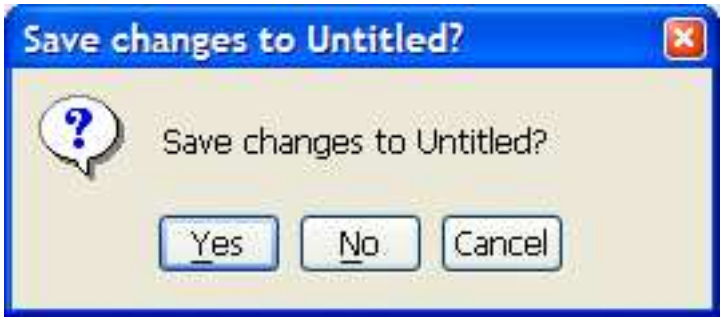
10.2.

10.3.

10.3.1.

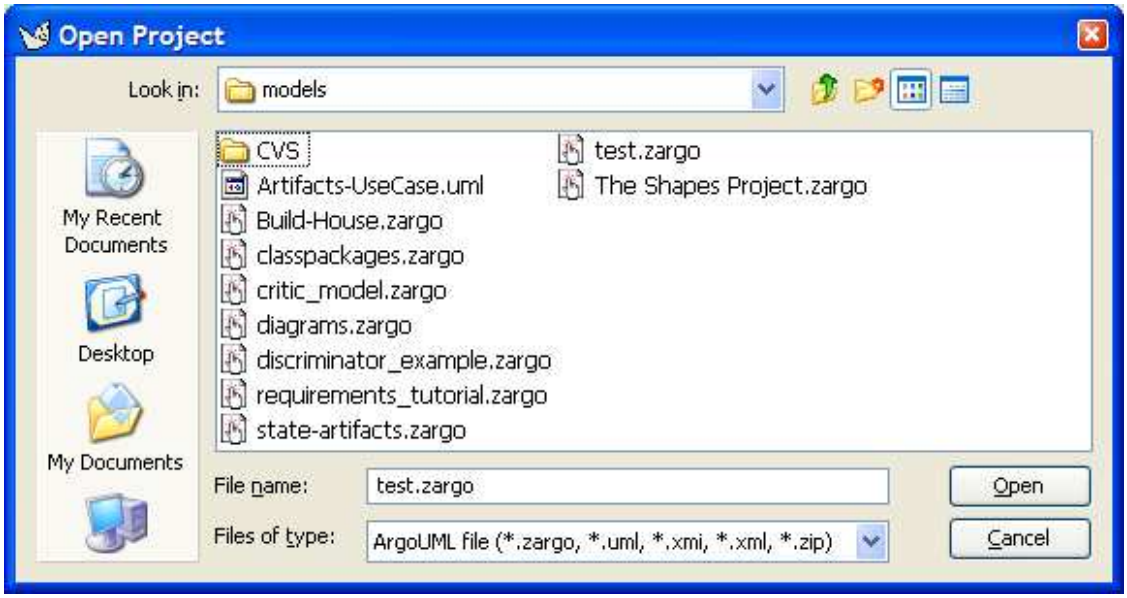
Atención

Figura 10.1.



10.3.2.

Figura 10.2.



-
-
-
-
-
-

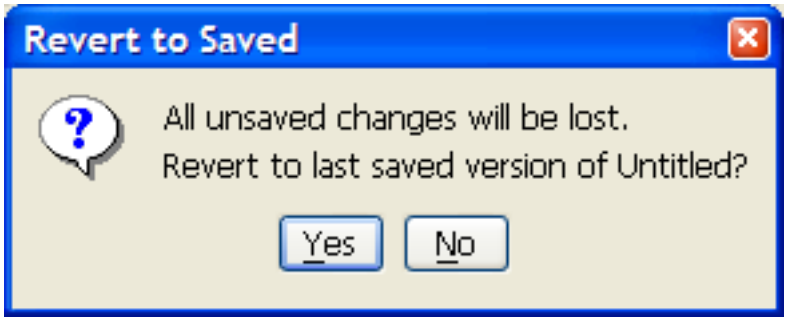
10.3.3.

Nota

10.3.4.

10.3.5.

Figura 10.3.



10.3.6.

Figura 10.4.

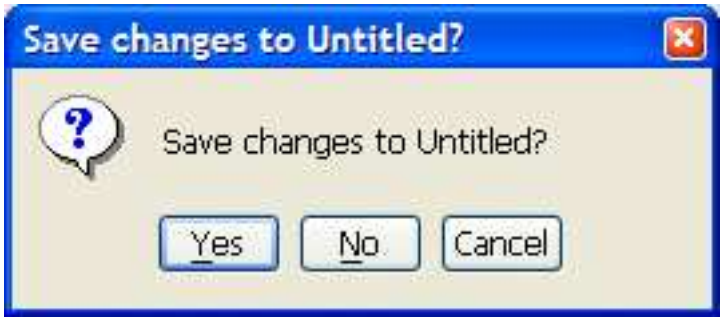
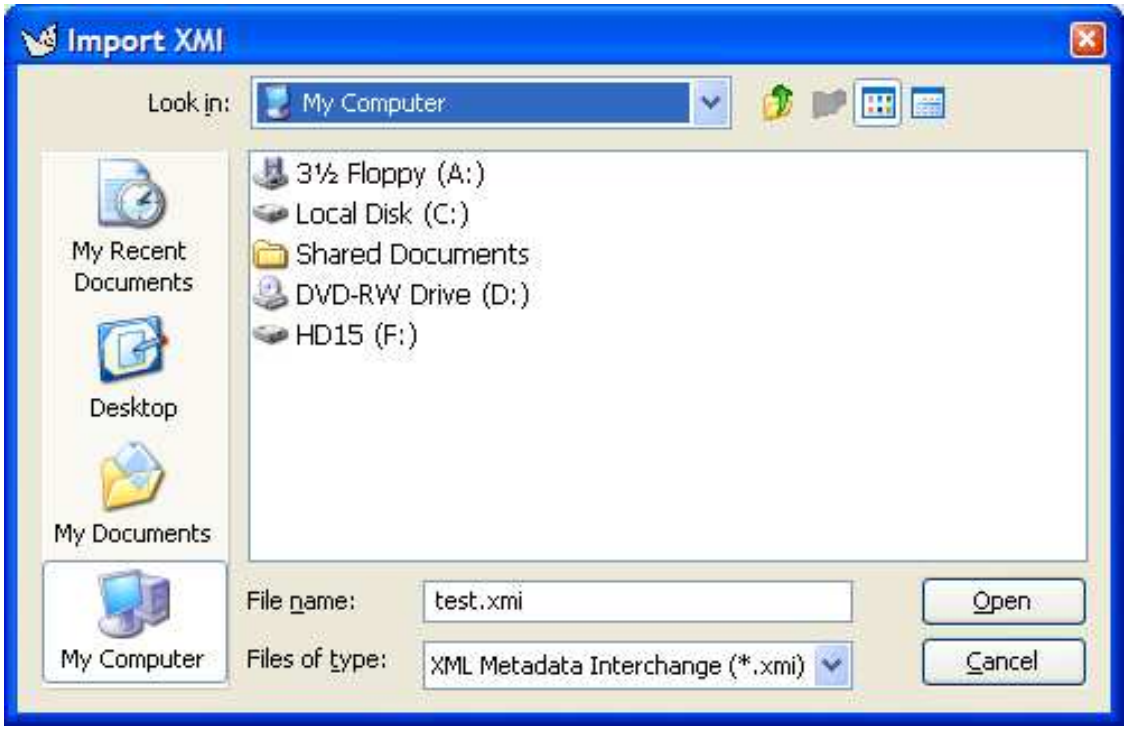
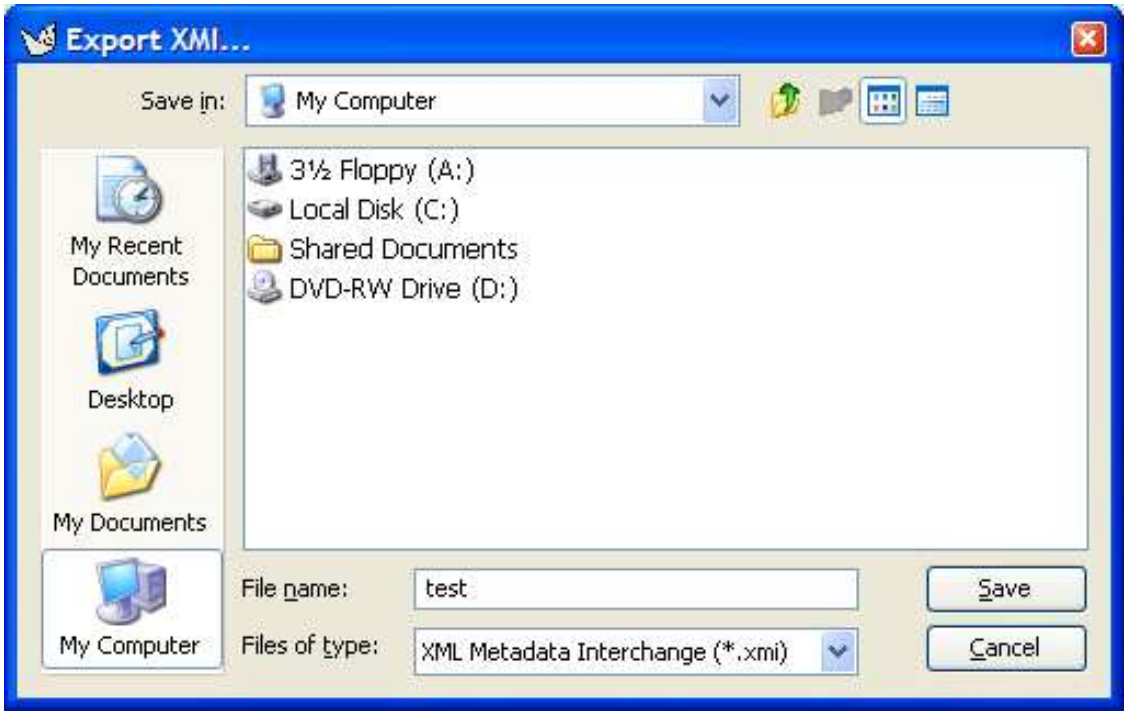


Figura 10.5.



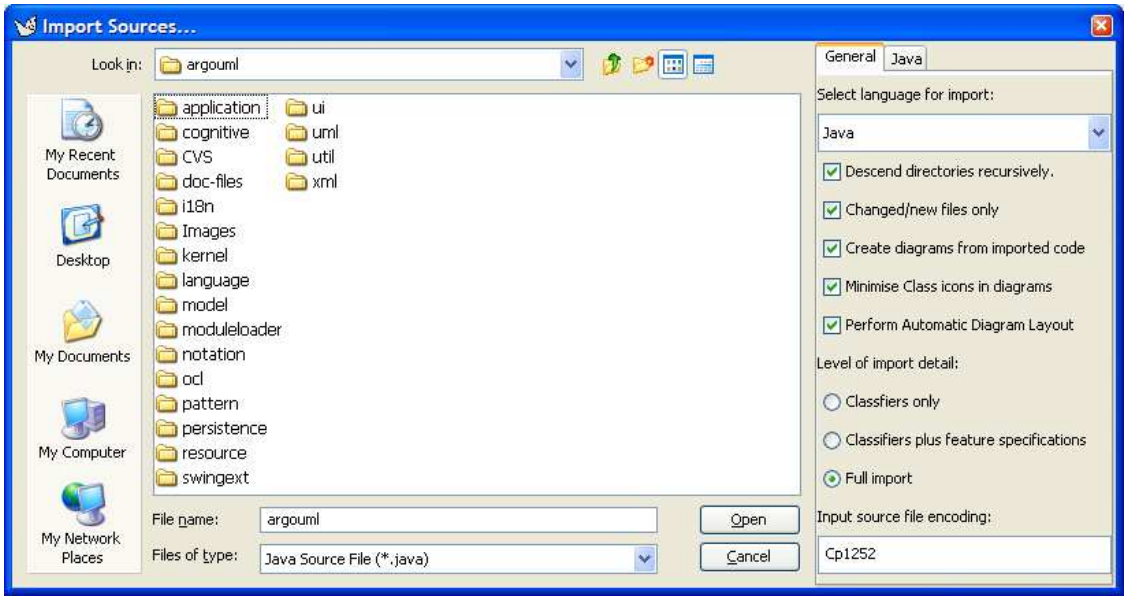
10.3.7.

Figura 10.6.



10.3.8.

Figura 10.7.



-
-
-
-

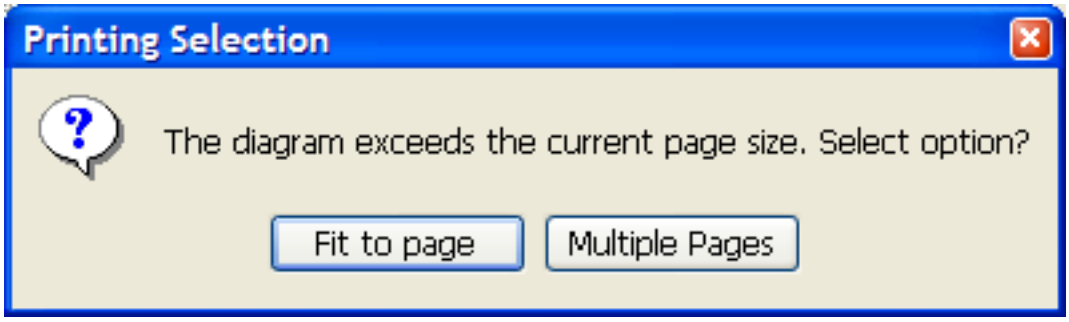
.
.
.

.
.

10.3.9.

10.3.10.

Figura 10.8.



Aviso

10.3.11.

.
.
.
.
.

10.3.12.

10.3.13.

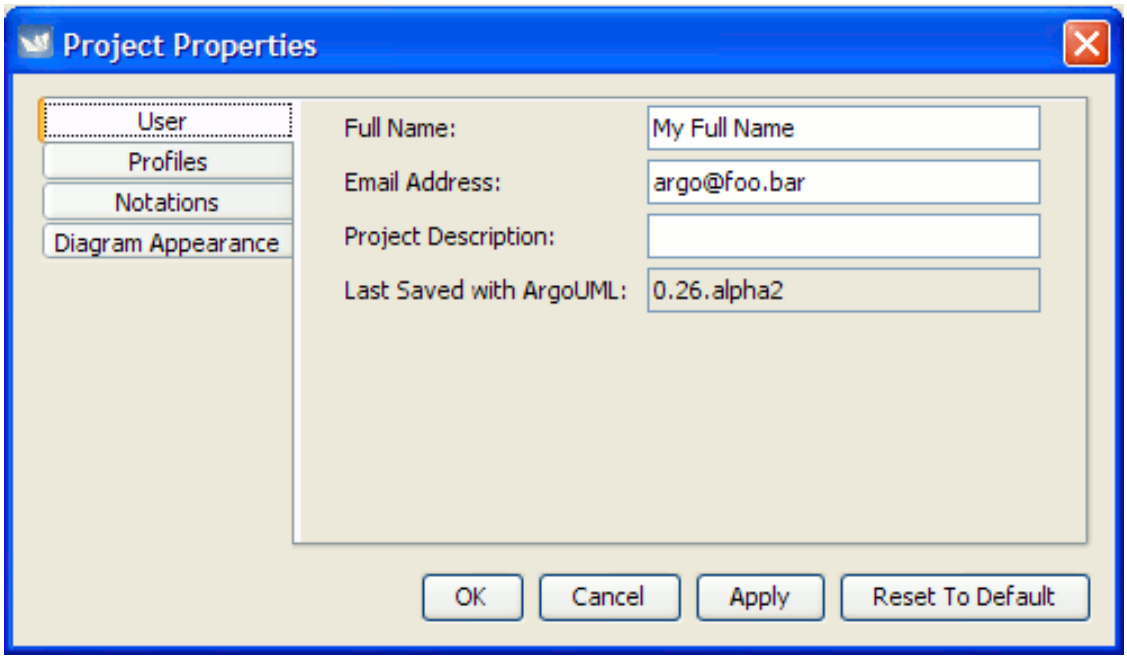
.
.

.
.

.
.
.

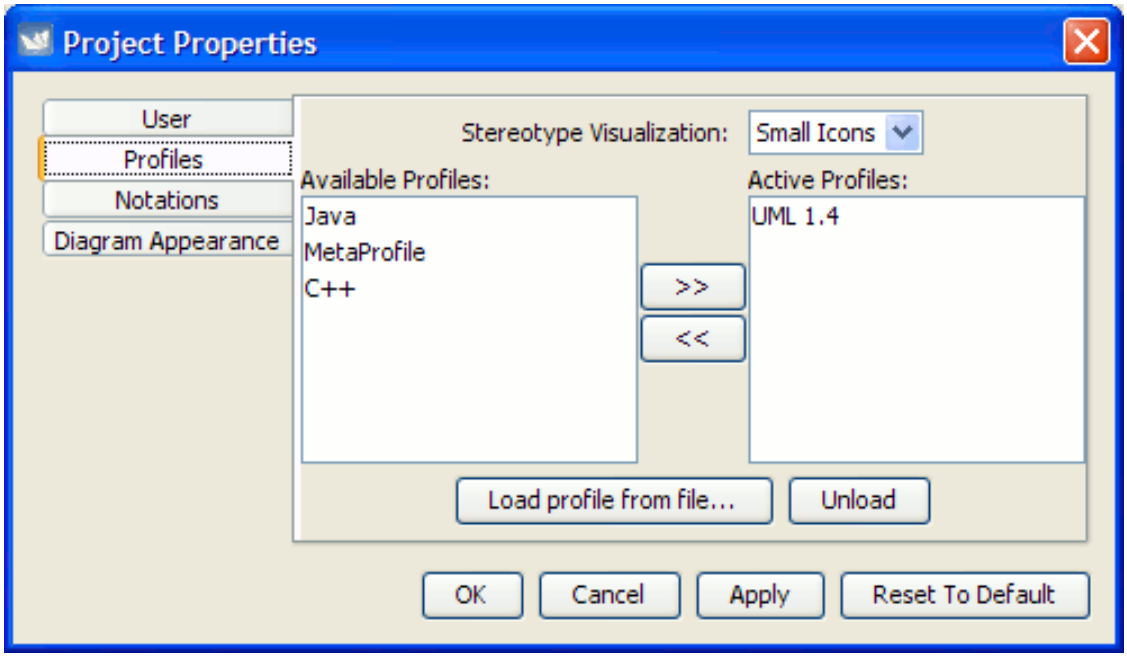
10.3.14.

Figura 10.9.



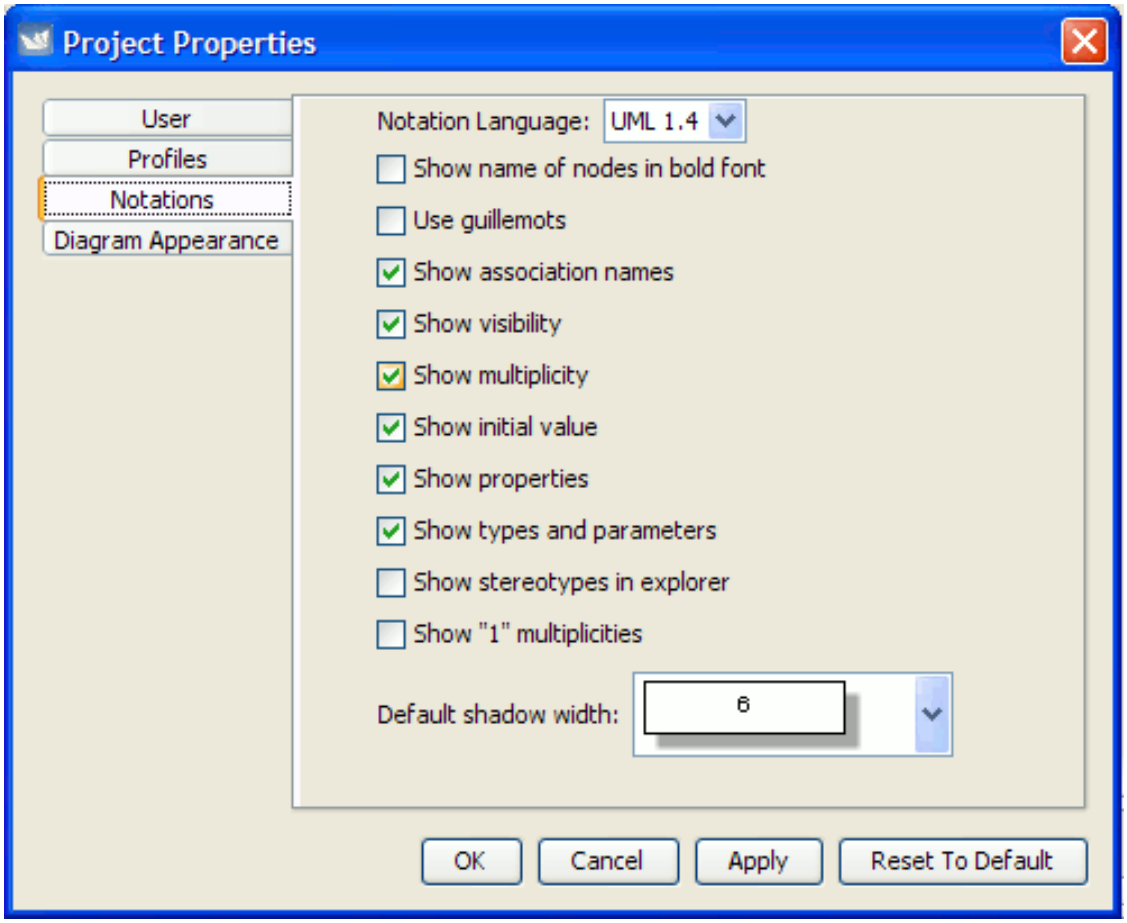
.
.
.

Figura 10.10.


Project Properties
User
Profiles
Notations
Diagram Appearance
Stereotype Visualization:
Small Icons
Available Profiles:
Java
MetaProfile
C++
>>
<<
Active Profiles:
UML 1.4
Load profile from file...
Unload
OK
Cancel
Apply
Reset To Default

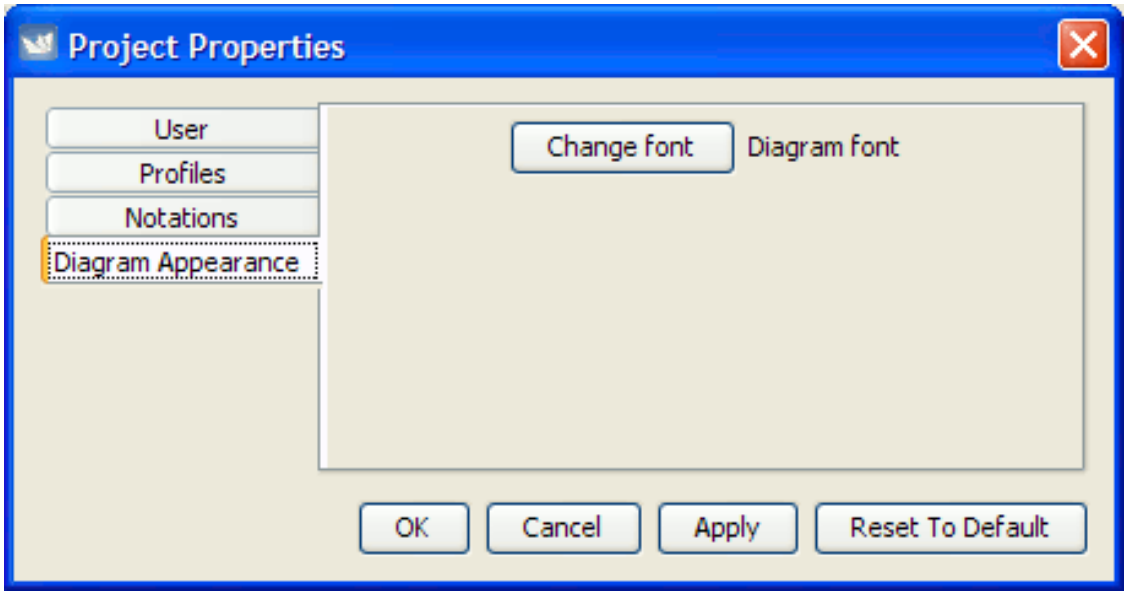
.
.

Figura 10.11.


Project Properties
User
Profiles
Notations
Diagram Appearance
Notation Language:
UML 1.4
Show name of nodes in bold font
Use guillemots
Show association names
Show visibility
Show multiplicity
Show initial value
Show properties
Show types and parameters
Show stereotypes in explorer
Show "1" multiplicities
Default shadow width:
6
OK
Cancel
Apply
Reset To Default

-
-
-
-
-
-
-
-
-

Figura 10.12.

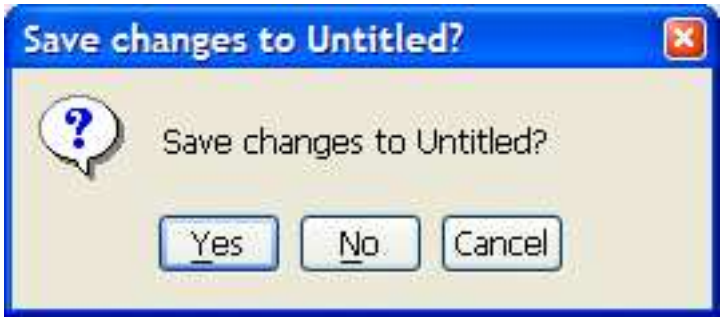


10.3.15.

10.3.16.

- .
- .
- .
- .

Figura 10.13.



10.4.

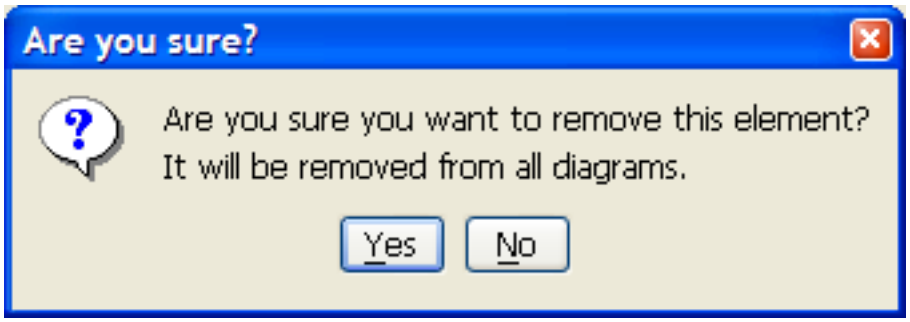
10.4.1.

- .
- .
- .
- .

10.4.2.

10.4.3.

Figura 10.14.

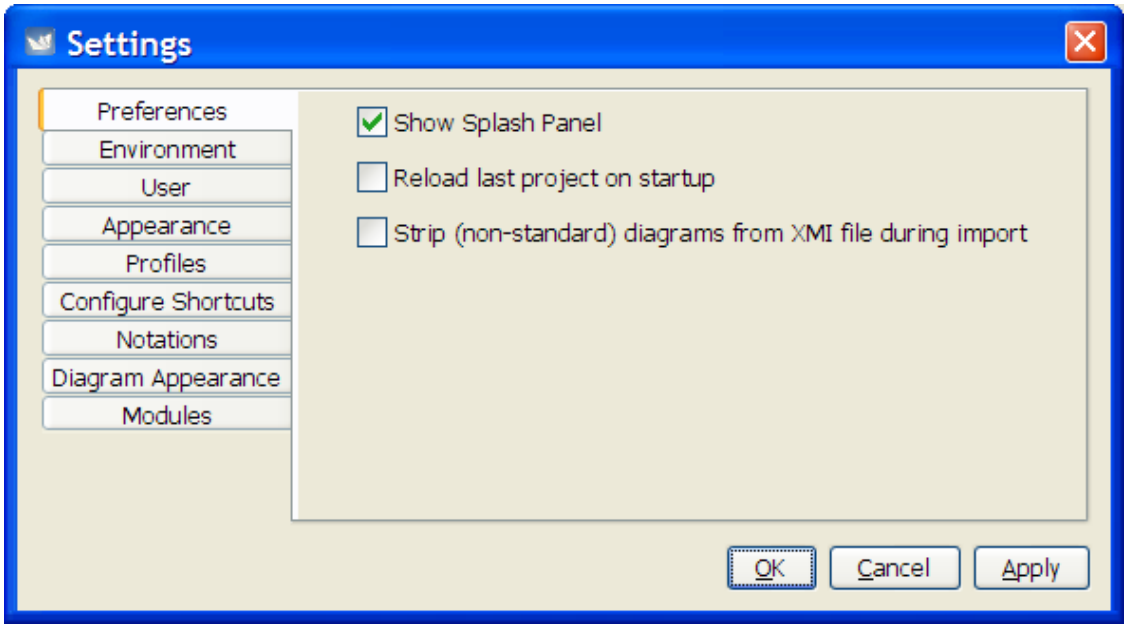


10.4.4.

10.4.5.

Sugerencia

Figura 10.15.



.

-
-

10.4.5.1.

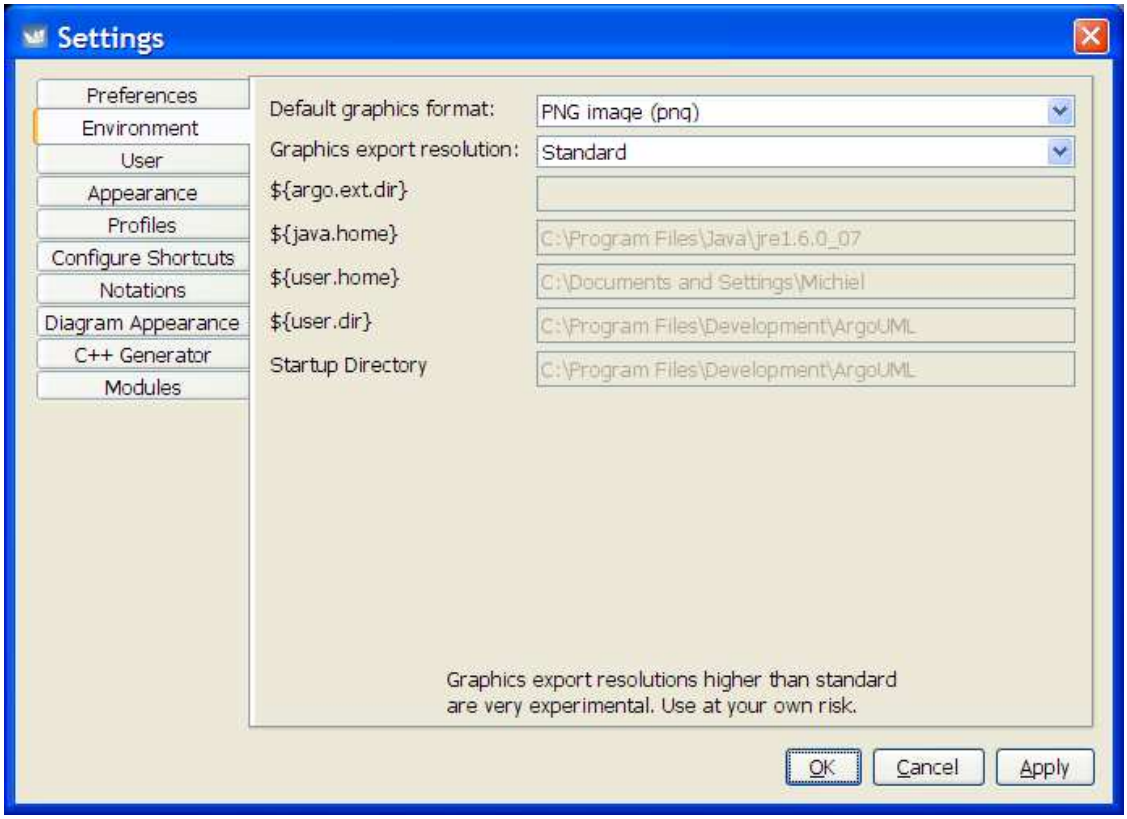
-

Sugerencia

-
-

10.4.5.2.

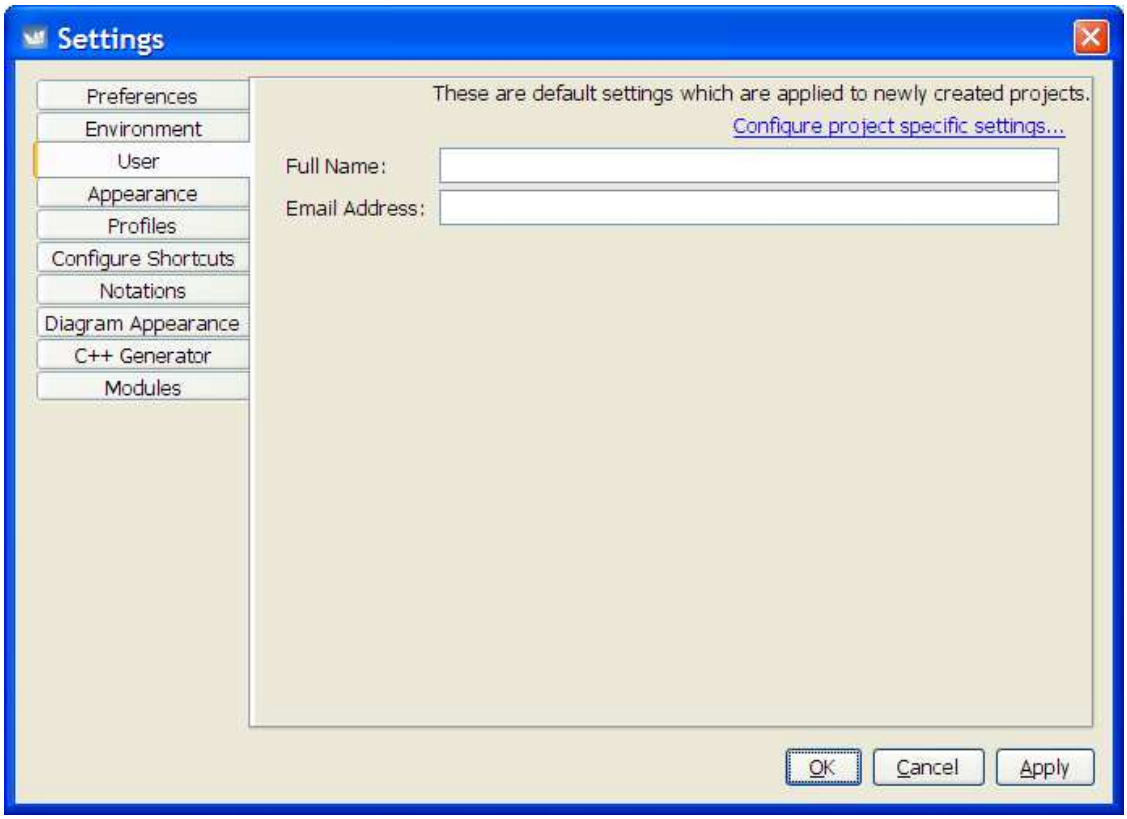
Figura 10.16.



-
-
-
-
-
-
-

10.4.5.3.

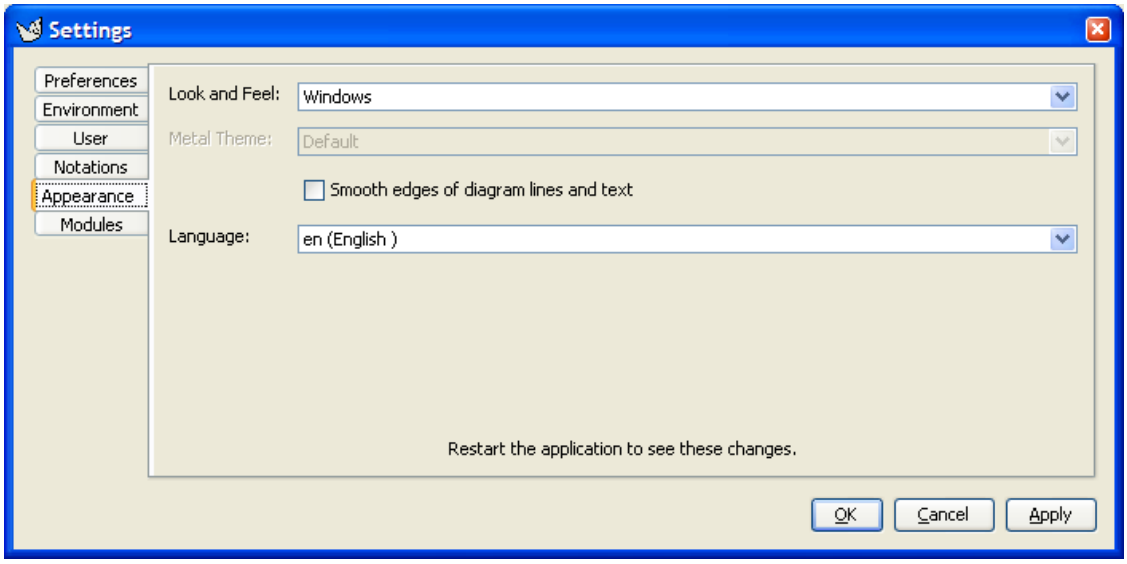
Figura 10.17.



.
.

10.4.5.4.

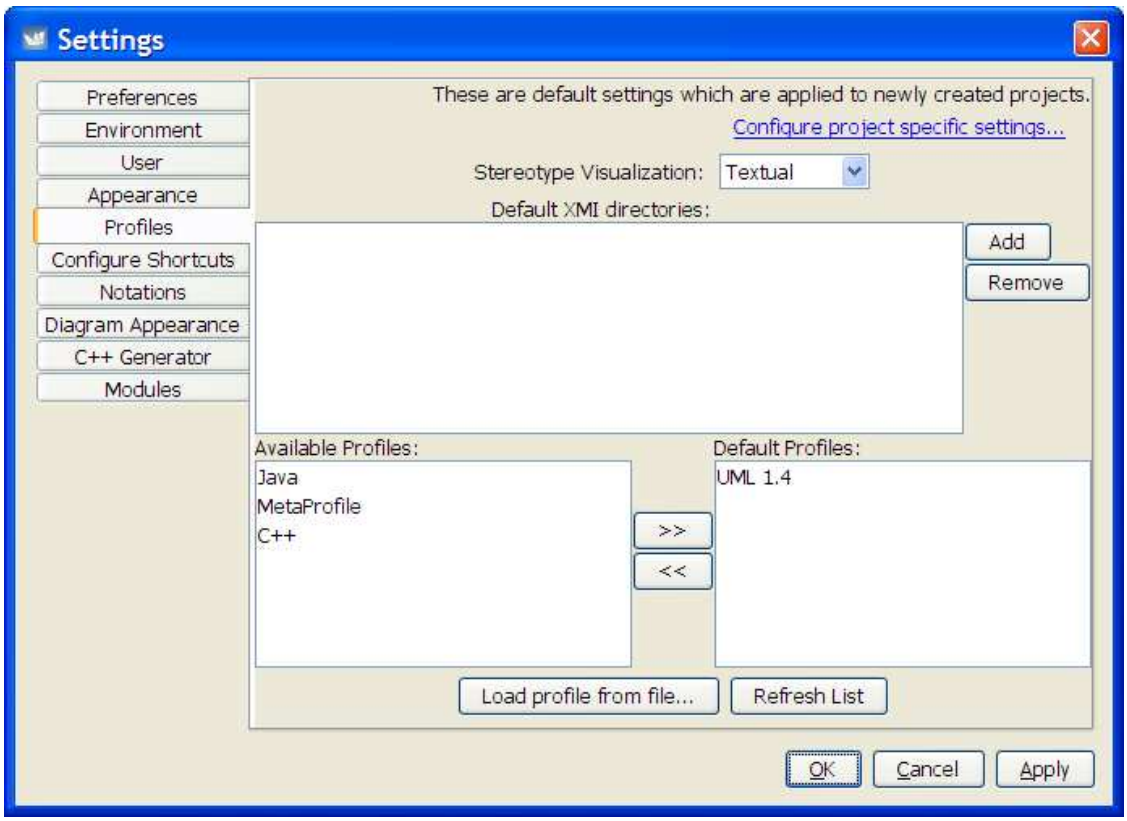
Figura 10.18.



.
.
.

10.4.5.5.

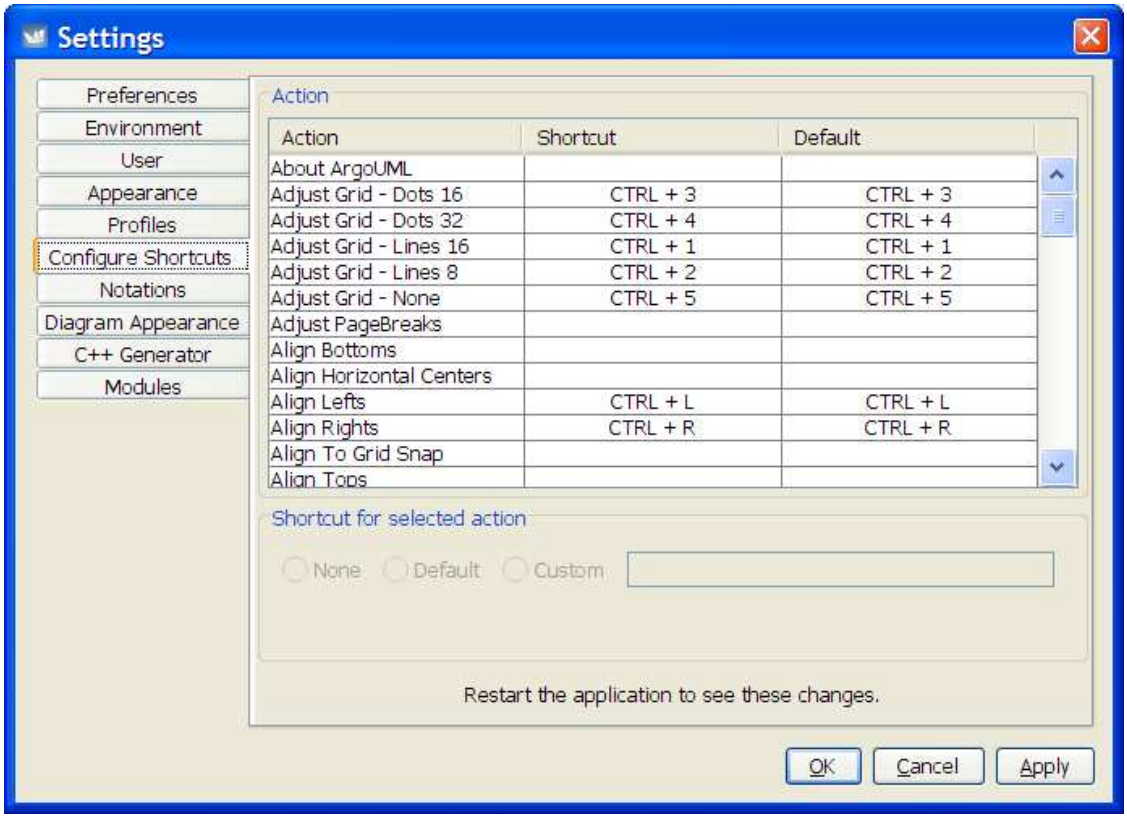
Figura 10.19.



.
.
.

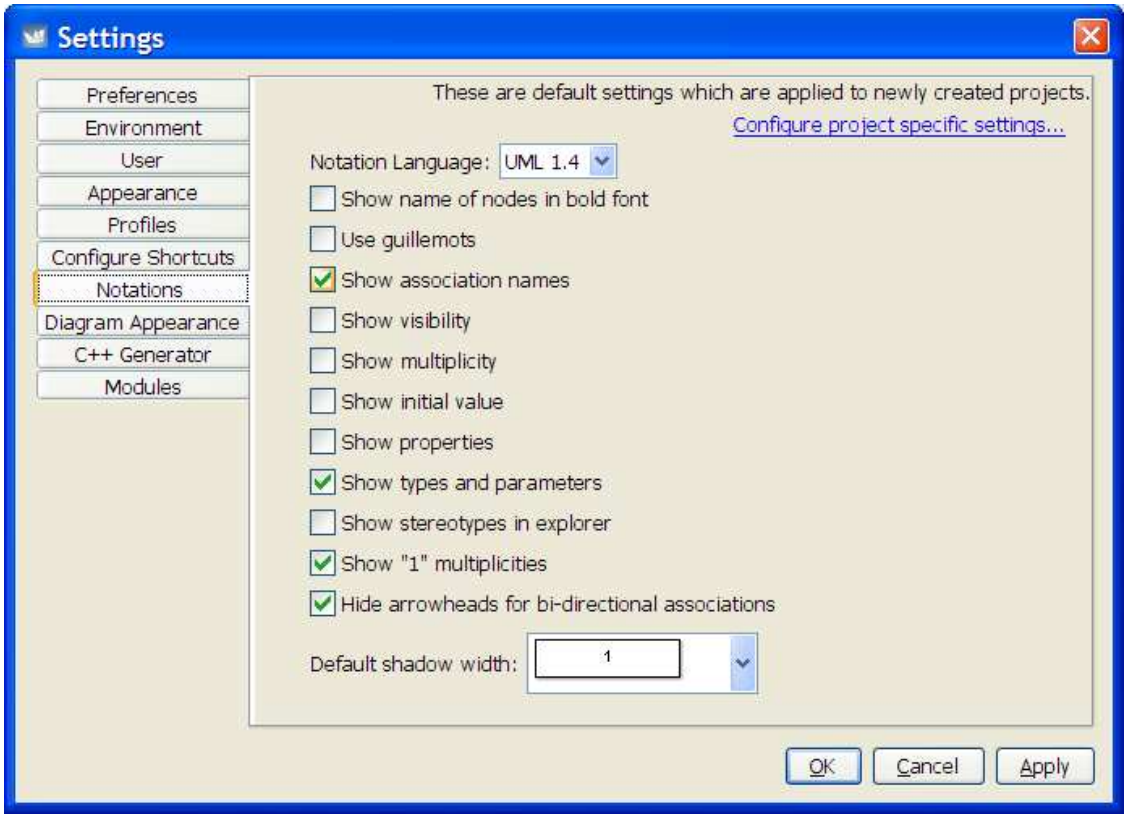
10.4.5.6.

Figura 10.20.



10.4.5.7.

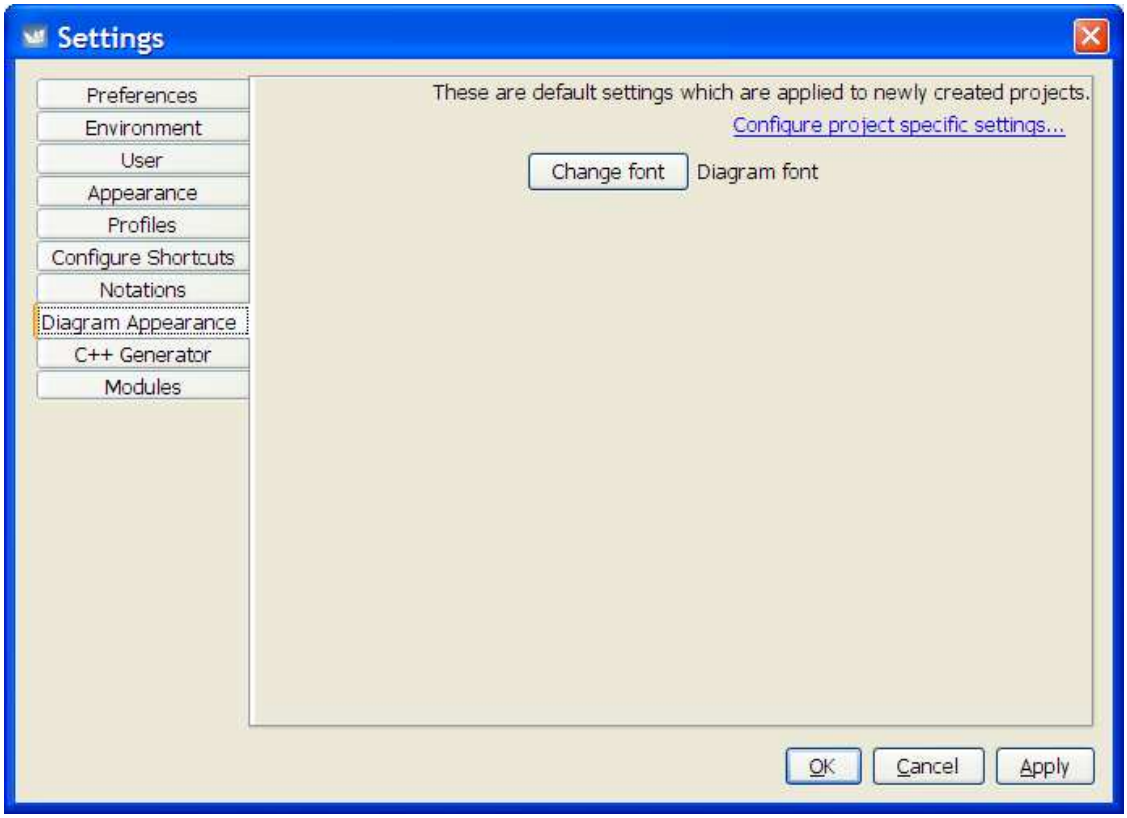
Figura 10.21.



-
-
-
-
-
-
-
-
-
-
-
-

10.4.5.8.

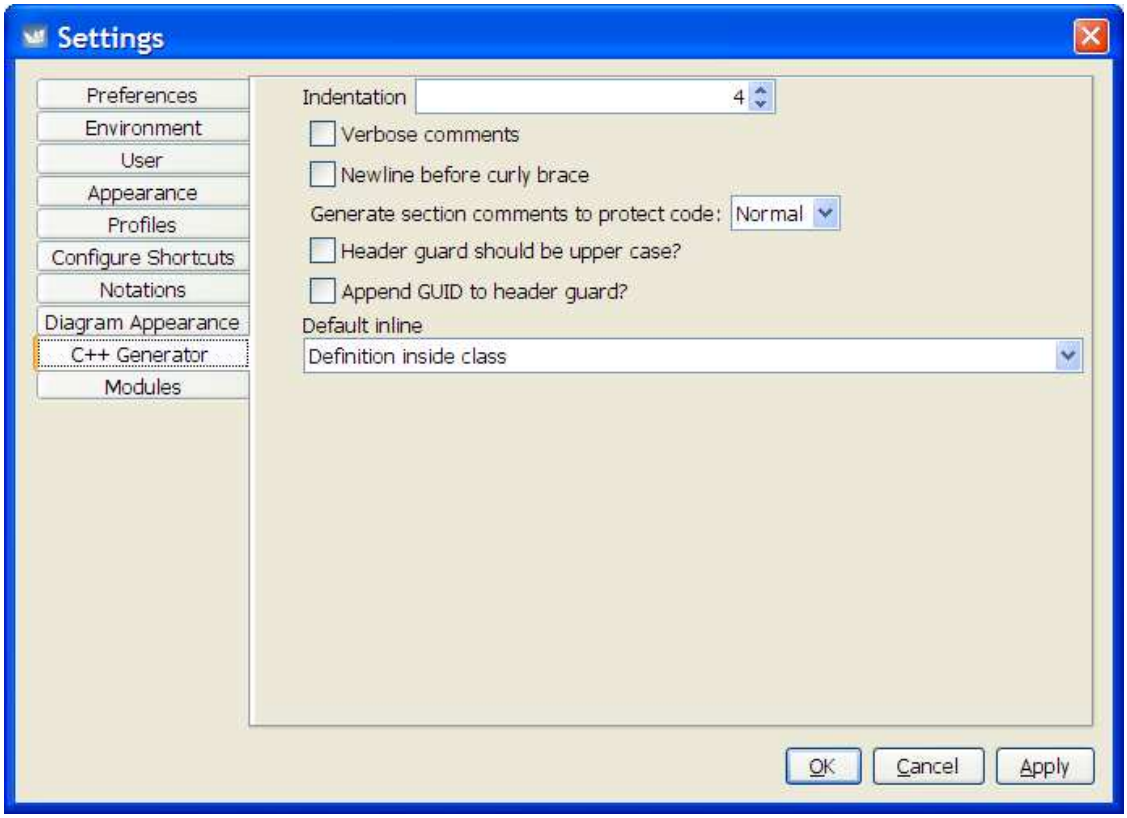
Figura 10.22.



10.4.5.9.

10.4.5.10.

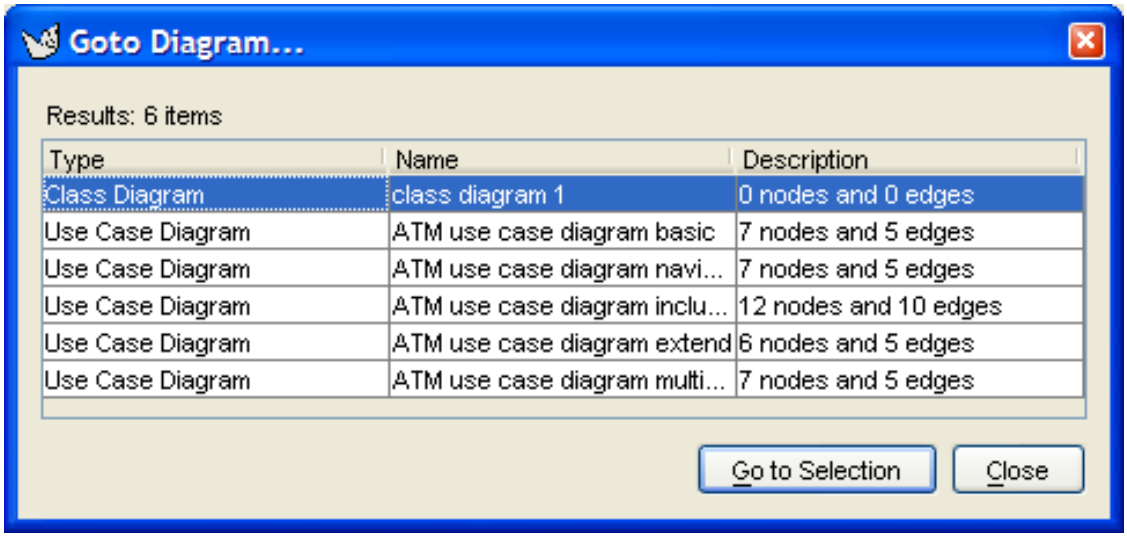
Figura 10.23.



10.5.

10.5.1.

Figura 10.24.

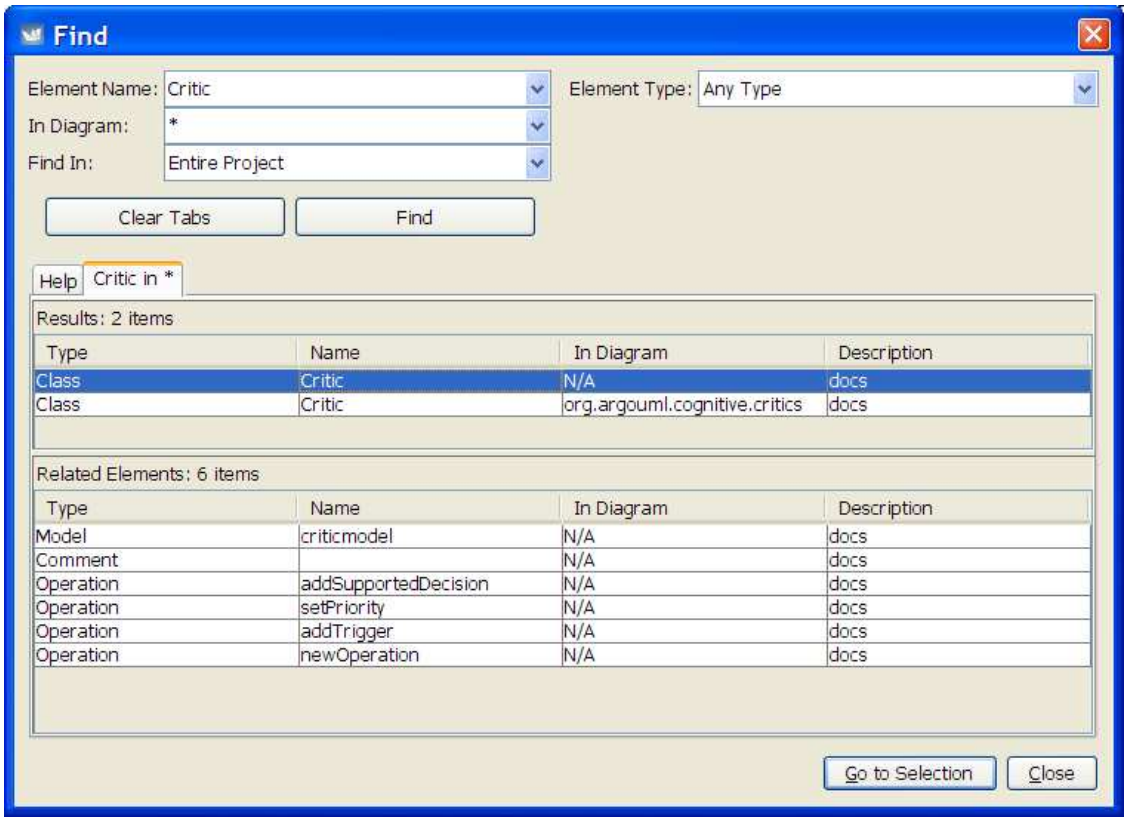


-
-
-

Aviso

10.5.2.

Figura 10.25.



-
-
-
-
-
-

-
-
-
-

Sugerencia

Aviso

10.5.3.

-
-
-

10.5.4.

-
-
-
-
-

10.5.5.

-
-
-
-

Nota

Nota

10.5.6.

Aviso

10.5.7.

10.5.8.

10.6.

10.6.1. 

Sugerencia

10.6.2.

Sugerencia

10.6.3.

10.6.4.

Sugerencia

10.6.5.

10.6.6.

10.6.7.

Sugerencia

10.7.

10.7.1.

-
-
-
-
-
-

Sugerencia

10.7.2.

-
-
-
-

10.7.3.

-

:
:
:

10.7.4.

10.7.5.

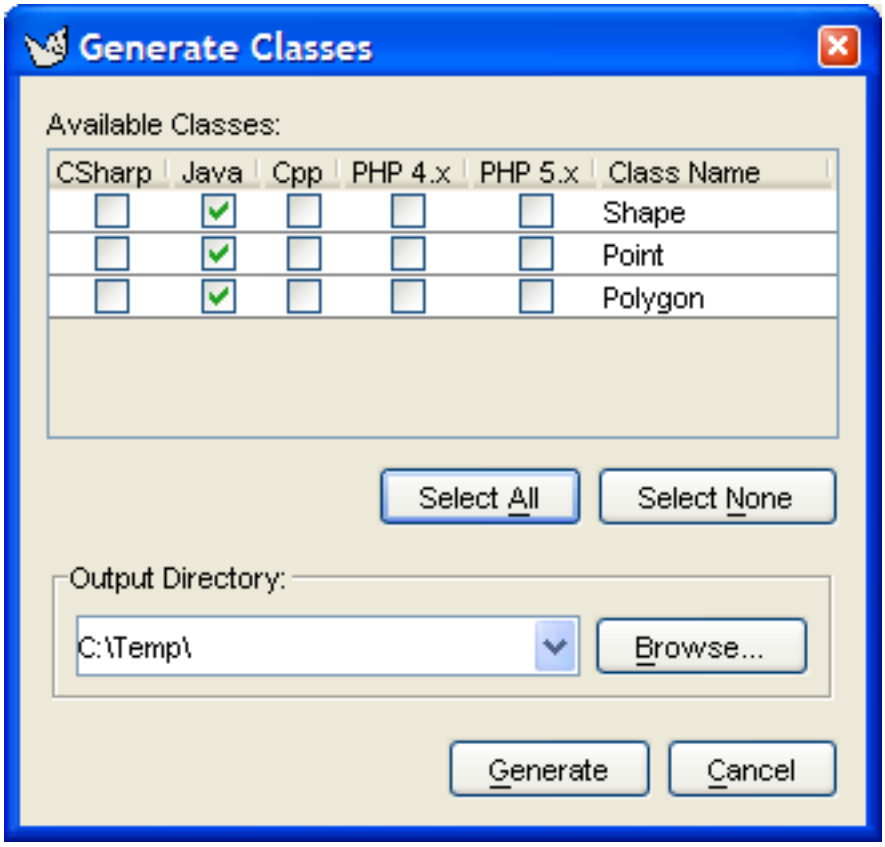
10.8.

Nota

Aviso

10.8.1.

Figura 10.26.



10.8.2.

10.8.3.

10.8.4.

10.9.

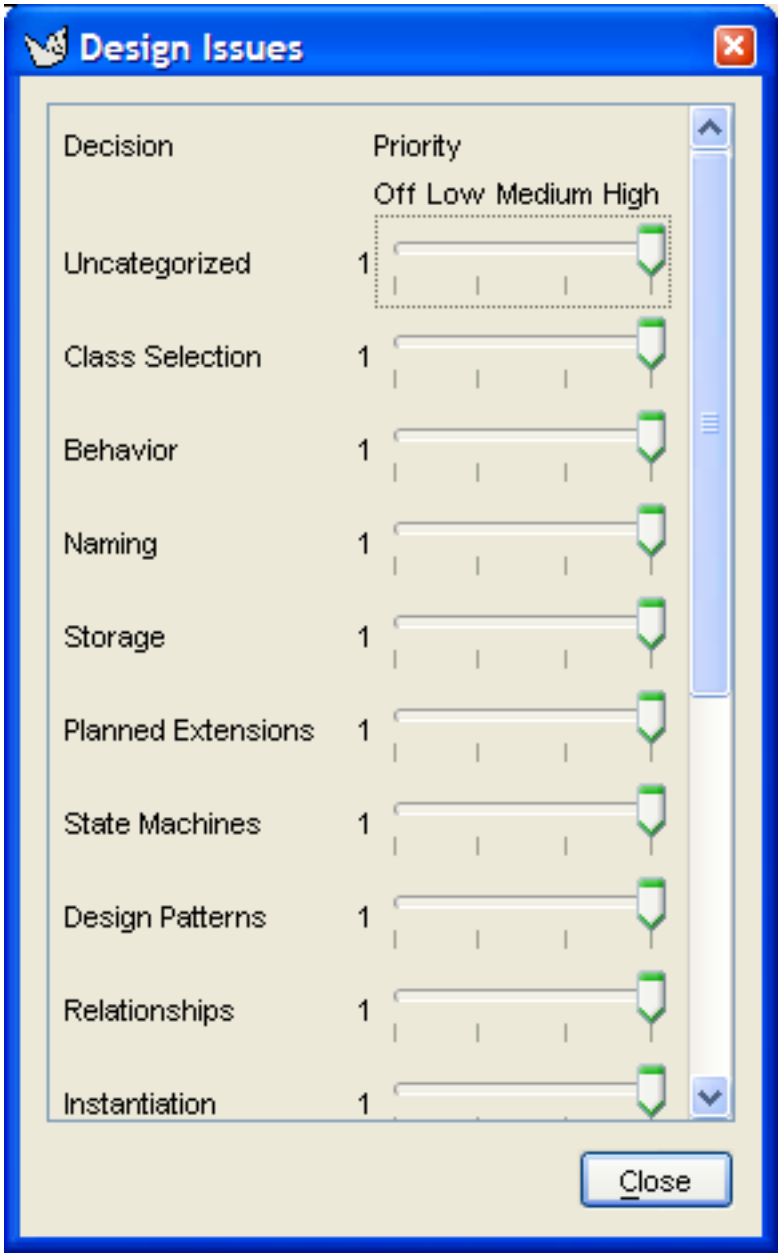
Nota

Nota

10.9.1.

10.9.2.

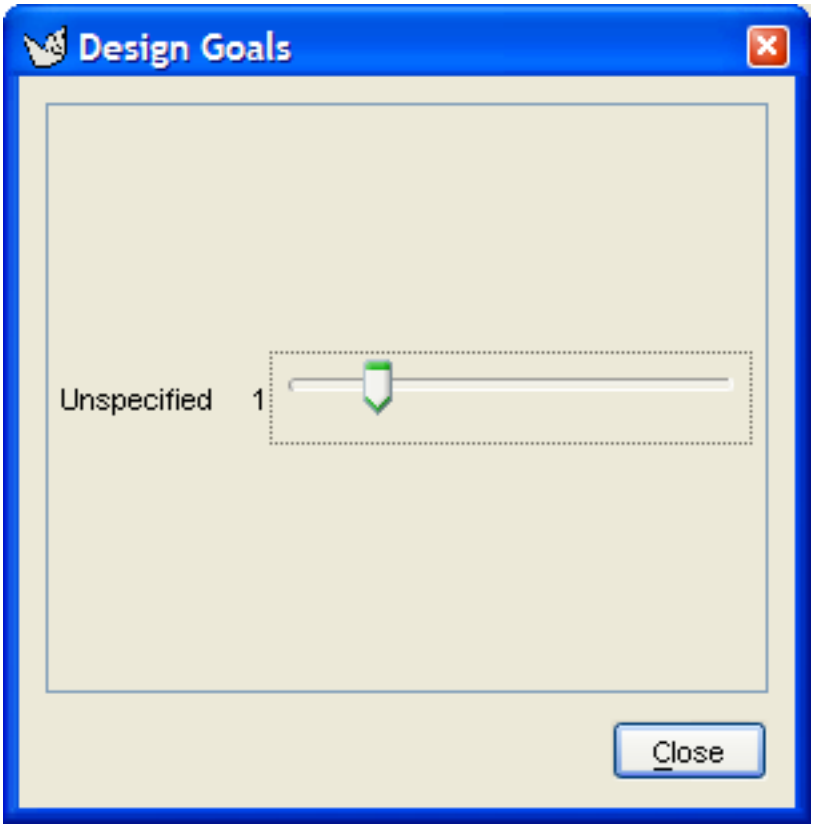
Figura 10.27.



Nota

10.9.3.

Figura 10.28.

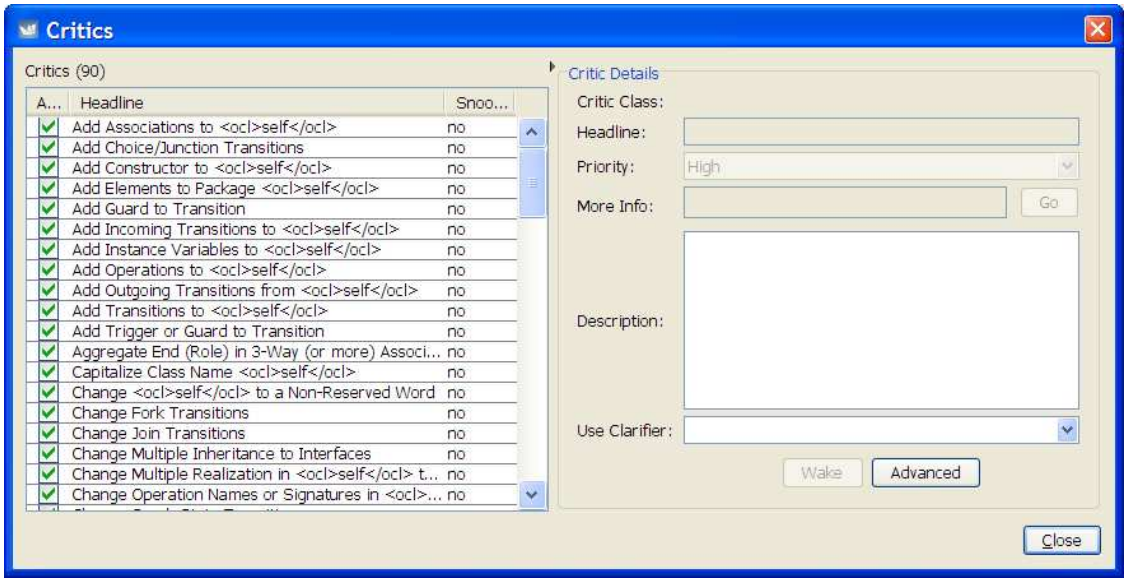


Sugerencia

Aviso

10.9.4.

Figura 10.29.



Nota

Aviso

Nota

Atención

- **Sugerencia**

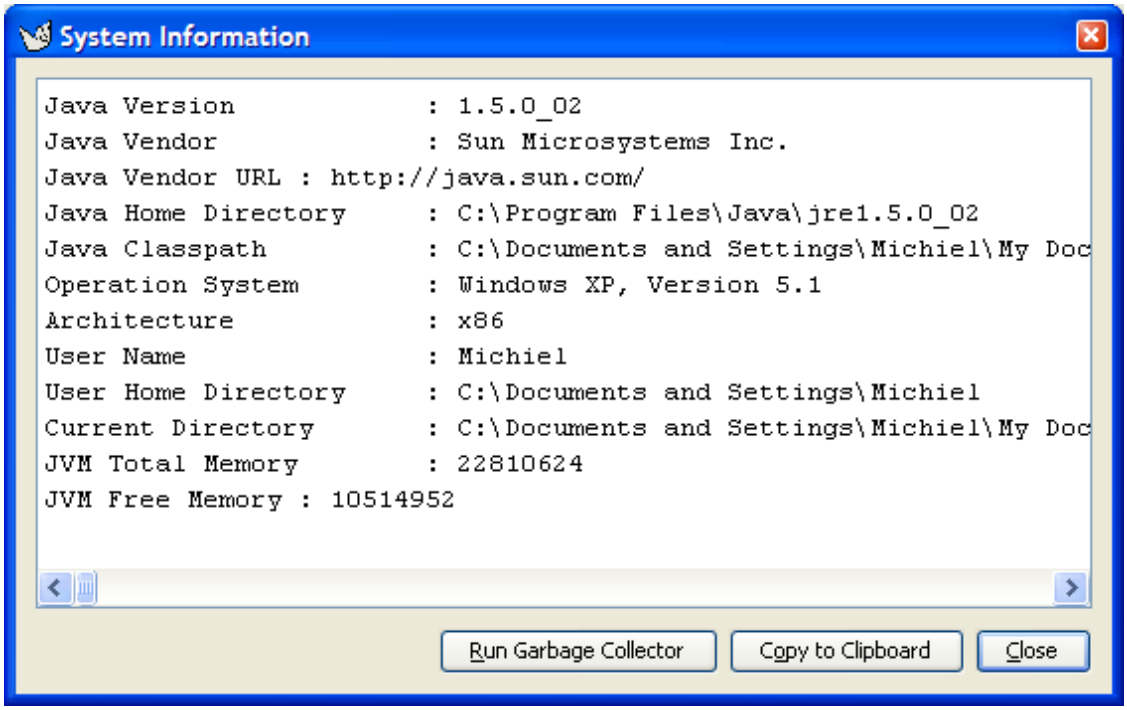
-

10.10.

10.11.

10.11.1.

Figura 10.30.



10.11.2.

Figura 10.31.


About ArgoUML
Splash
Version
Credits
Contact Info
Report bugs
Legal
vPRE-0.20.beta1
ArgoUML

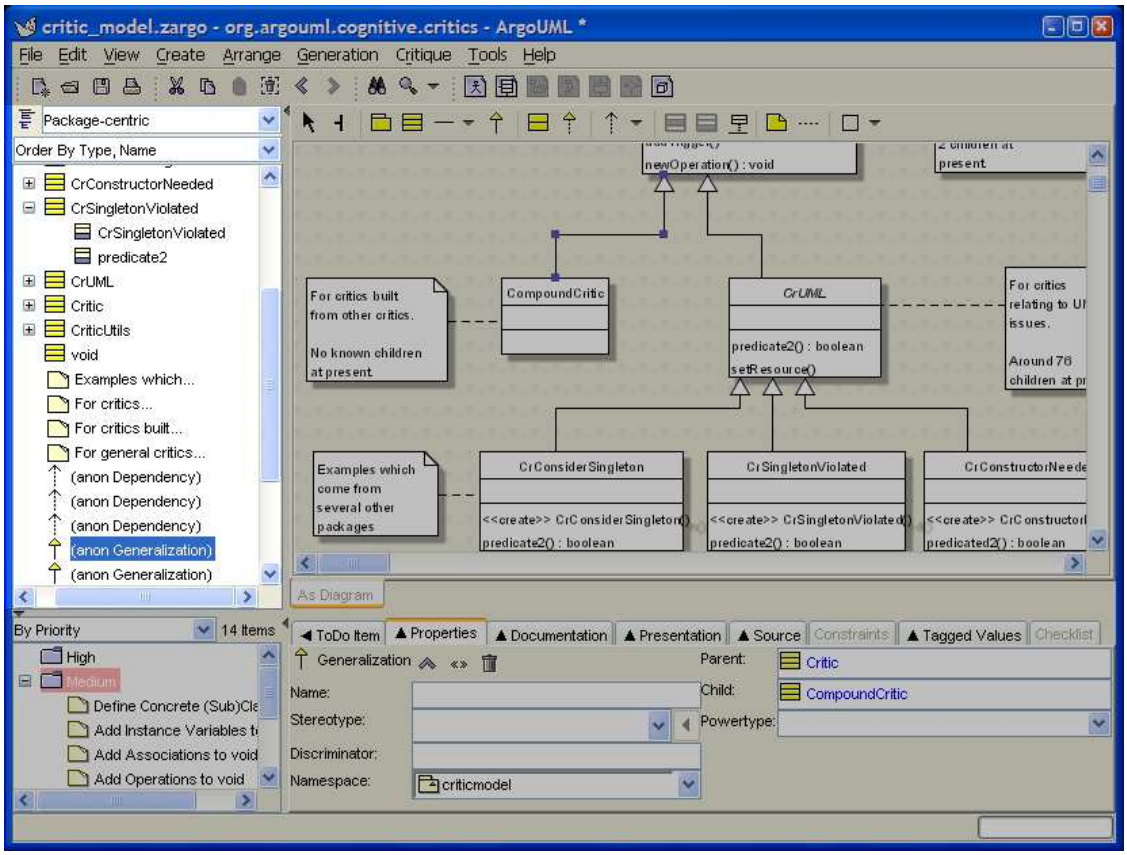
-
-
-
-
-
-

Atención

Capítulo 11.

11.1.

Figura 11.1.



11.2.

11.2.1.

Nota

11.2.2.

11.2.3.

11.2.3.1.

11.2.3.2.

11.2.4.

11.2.5.

11.3.

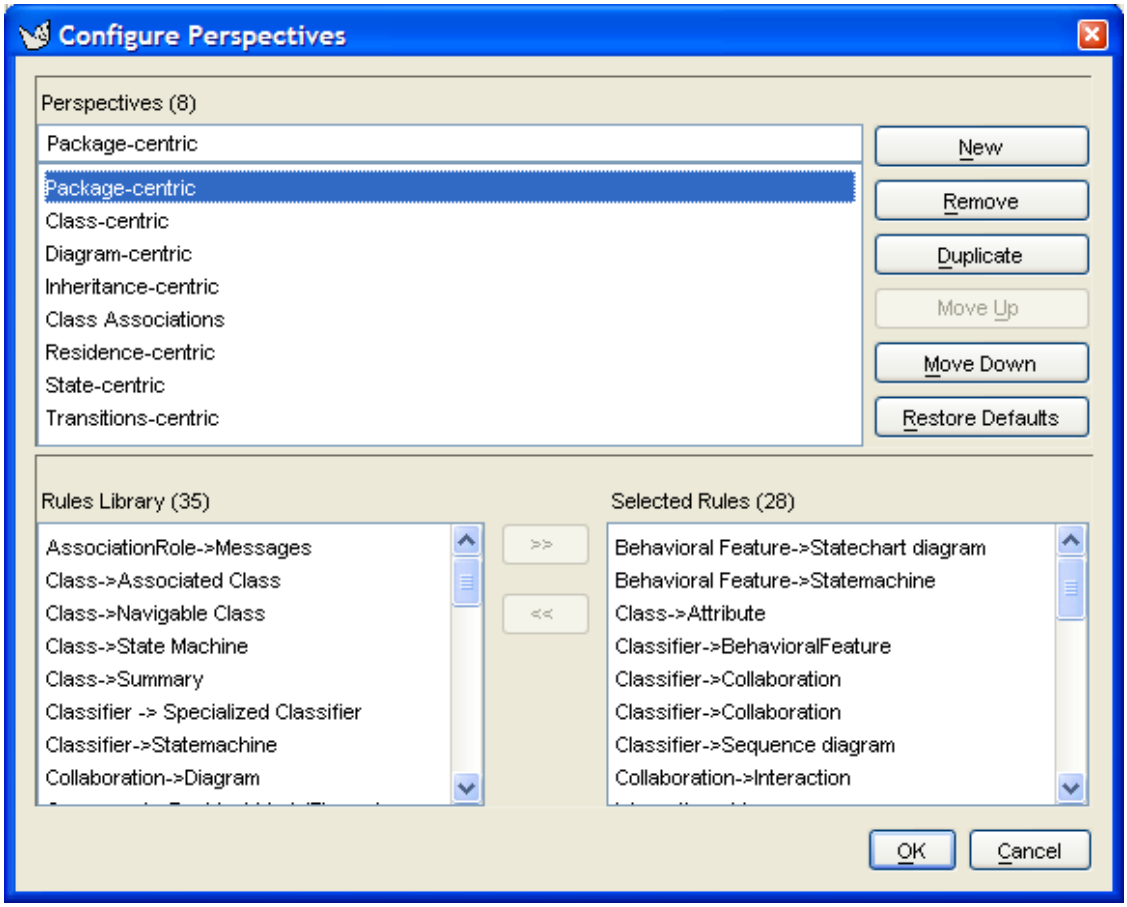
11.4.

- .
- .
- .
- .
- .
- .
- .
- .
- .

11.5.

11.5.1.

Figura 11.2.



-
-
-
-
-
-

11.6.

11.6.1.

11.6.2.

11.6.3.

Sugerencia

11.6.4.

Atención

11.6.5.

Aviso

Atención

:
:

11.6.6.

11.6.7.

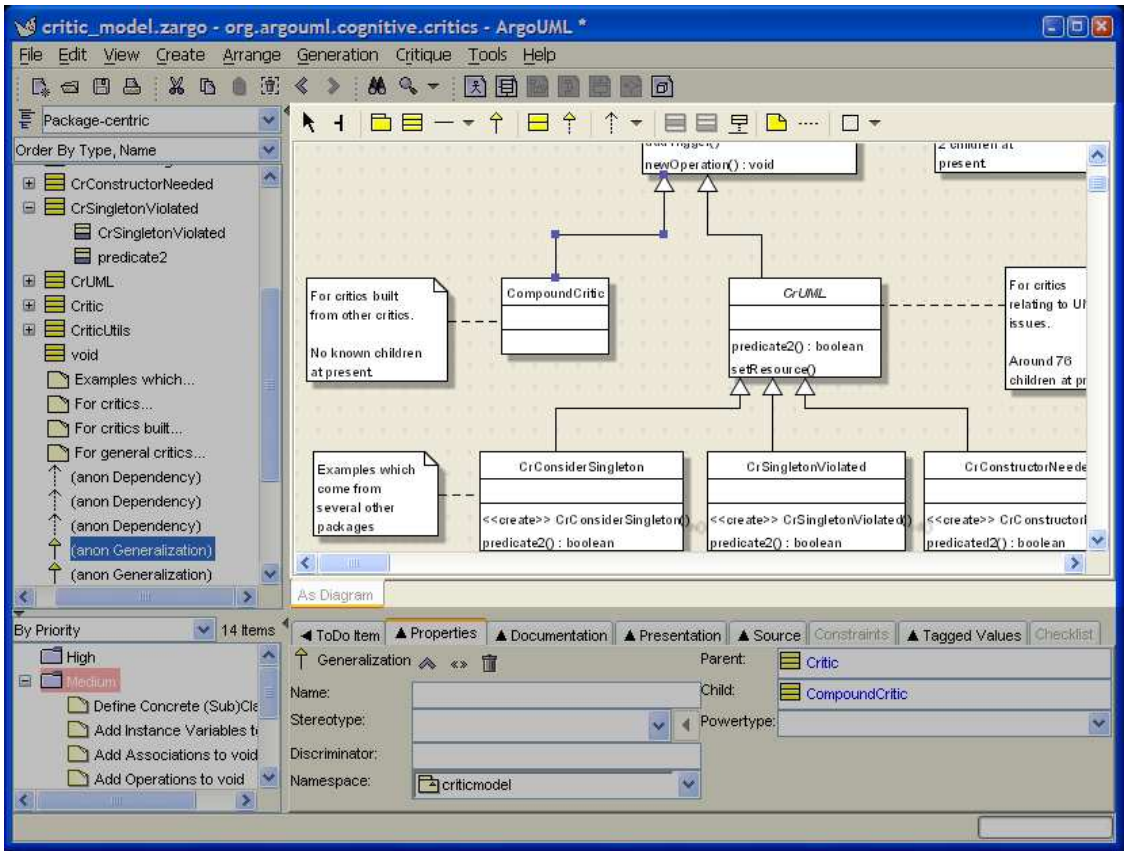
11.6.8.

11.6.9.

Capítulo 12.

12.1.

Figura 12.1.

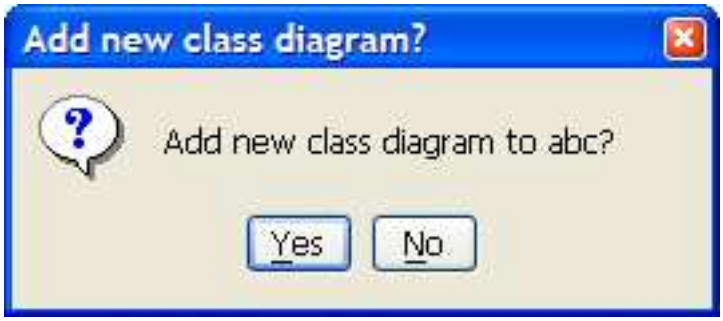


12.2.

12.2.1.

12.2.2.

Figura 12.2.



12.2.3.

12.2.4.

12.2.5.

12.2.6.

12.2.7.

12.2.8.

12.3.

12.3.1.

12.3.2.

12.4.

.
.
.
.

12.4.1.

.
.

Sugerencia

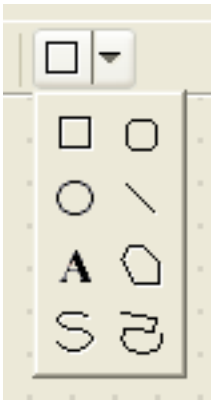
12.4.2.

Atención

12.4.3.

Sugerencia

Figura 12.3.

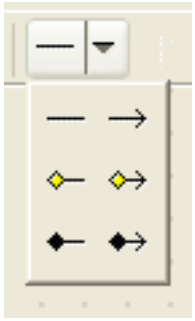


-
-
-
-
-
-
-
-

12.4.4.

-
-
-

Figura 12.4.



-
-
-
-
-

Nota

12.4.5.

-
-
-
-
-
-
-
-
-
-

Nota

-

Nota

-
-

12.4.6.

-
-
-
-
-
-
-

12.4.7.

-
-
-
-
-

Nota

12.4.8.

- **Atención**
- **Atención**
- **Atención**
- **Atención**
- **Atención**

12.4.9.

- **Atención**
- **Atención**
- **Atención**
- **Atención**

12.4.10.

Nota

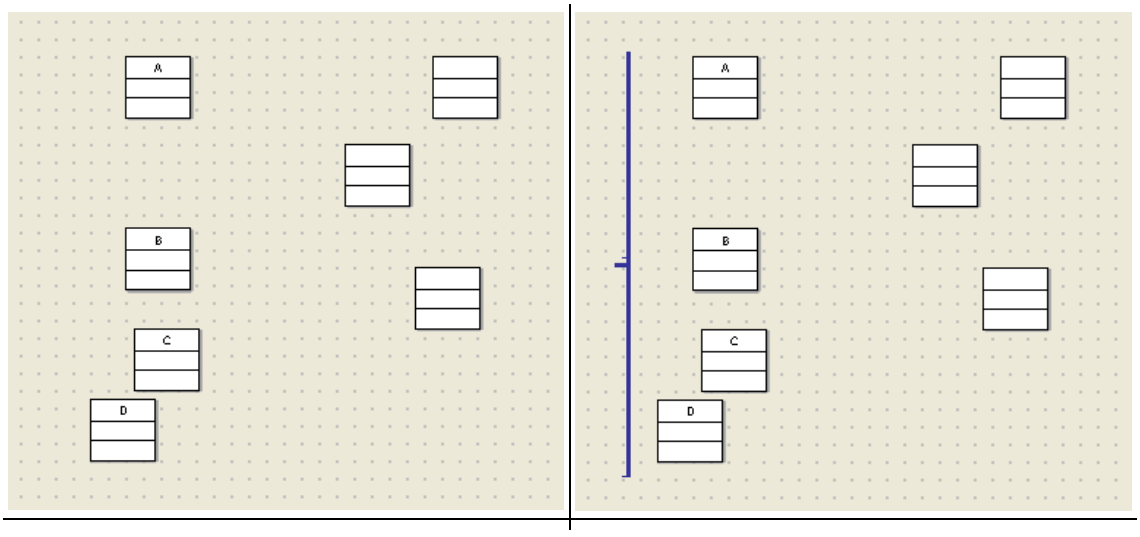
-
-
-
-
-
-
-
-

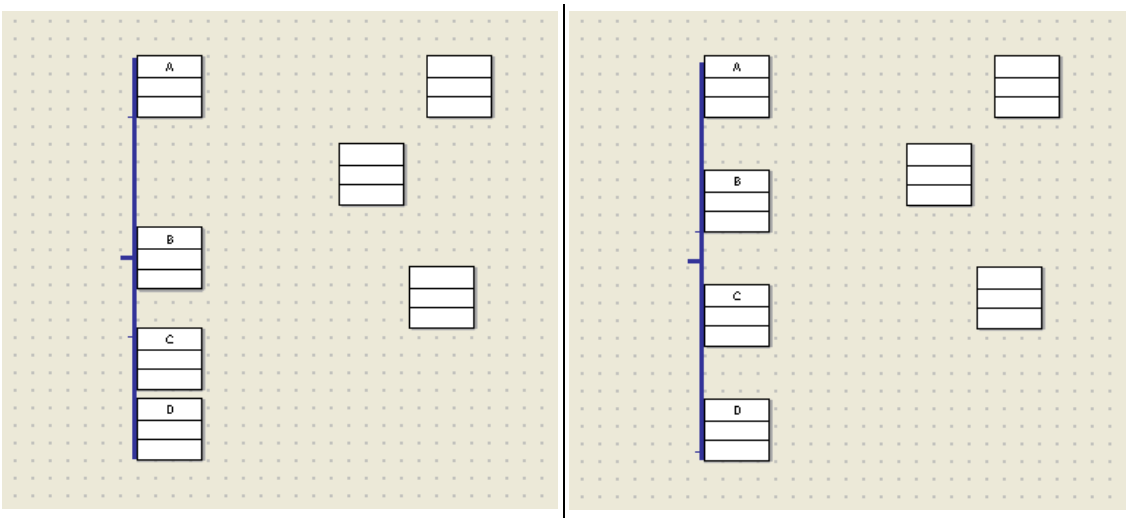
Atención

-
-

12.5.

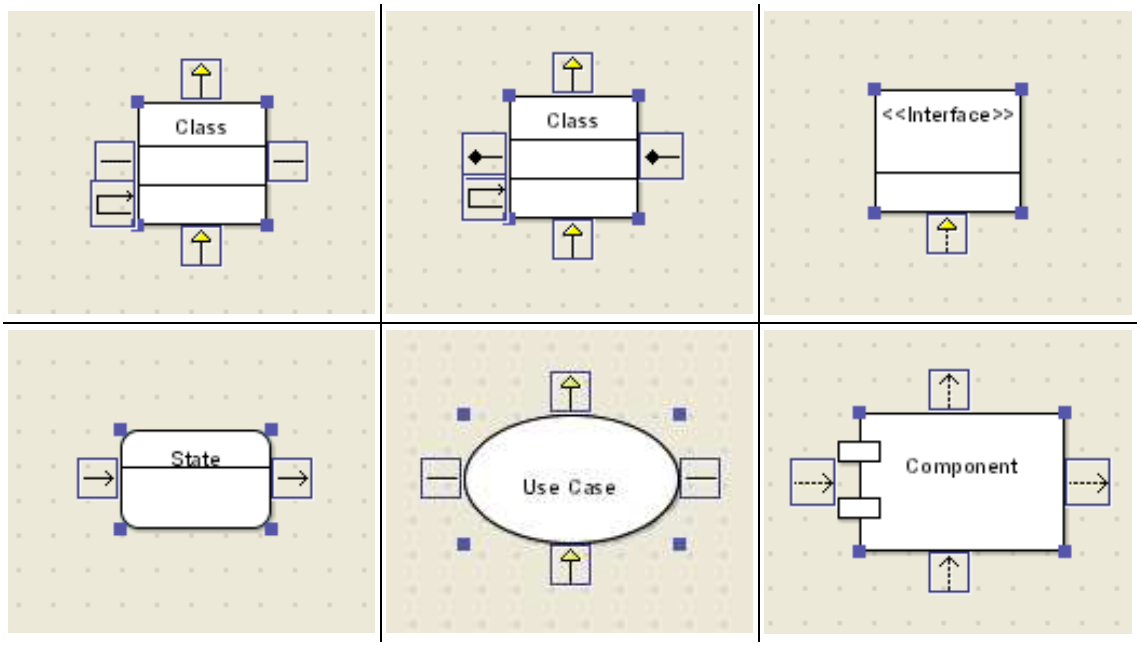
Figura 12.5.





12.6.

Figura 12.6.



12.7.

:

.

12.8.

12.9.

12.10.

12.10.1.

12.10.2.

.
.
.
.

12.10.3.

.
.
.
.
.

12.10.4.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

12.10.5.

.
.
.
.

Nota

12.10.6.

.
.
.
.

12.10.7.

.
.
.

Atención

12.10.8.

.
.
.

Nota

Nota

12.11.

12.11.1.

Figura 12.7.

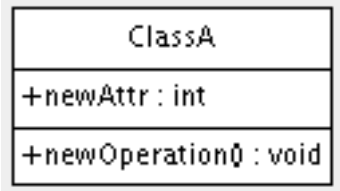
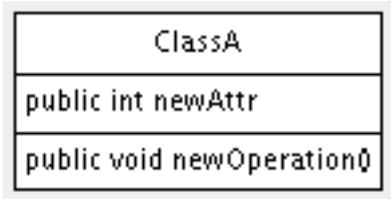
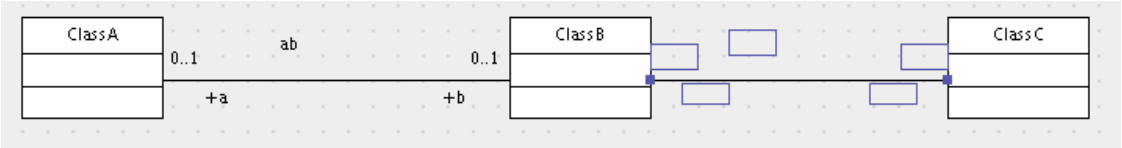


Figura 12.8.



12.11.2.

Figura 12.9.

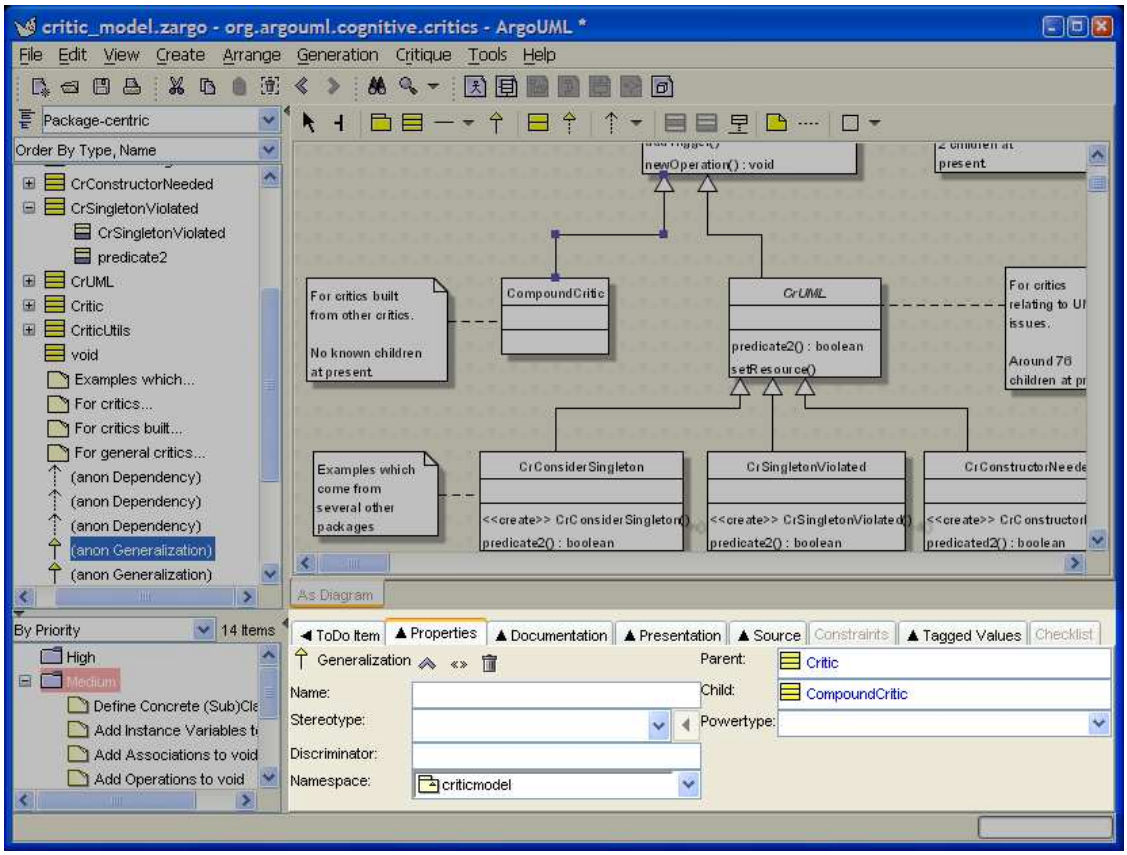


12.11.3.

Capítulo 13.

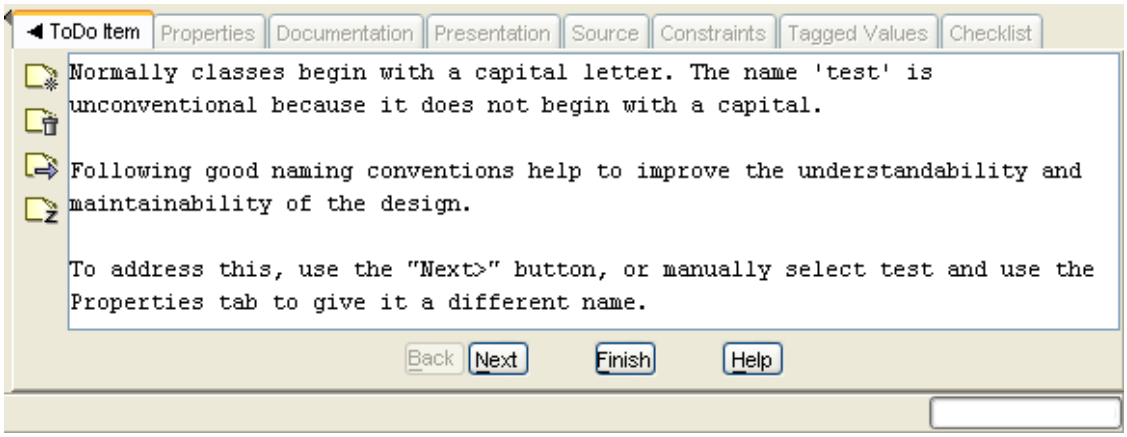
13.1.

Figura 13.1.



13.2.

Figura 13.2.


ToDo Item
Properties
Documentation
Presentation
Source
Constraints
Tagged Values
Checklist
Normally classes begin with a capital letter. The name 'test' is unconventional because it does not begin with a capital.
Following good naming conventions help to improve the understandability and maintainability of the design.
To address this, use the "Next>" button, or manually select test and use the Properties tab to give it a different name.
Back
Next
Finish
Help

•

Figura 13.3.

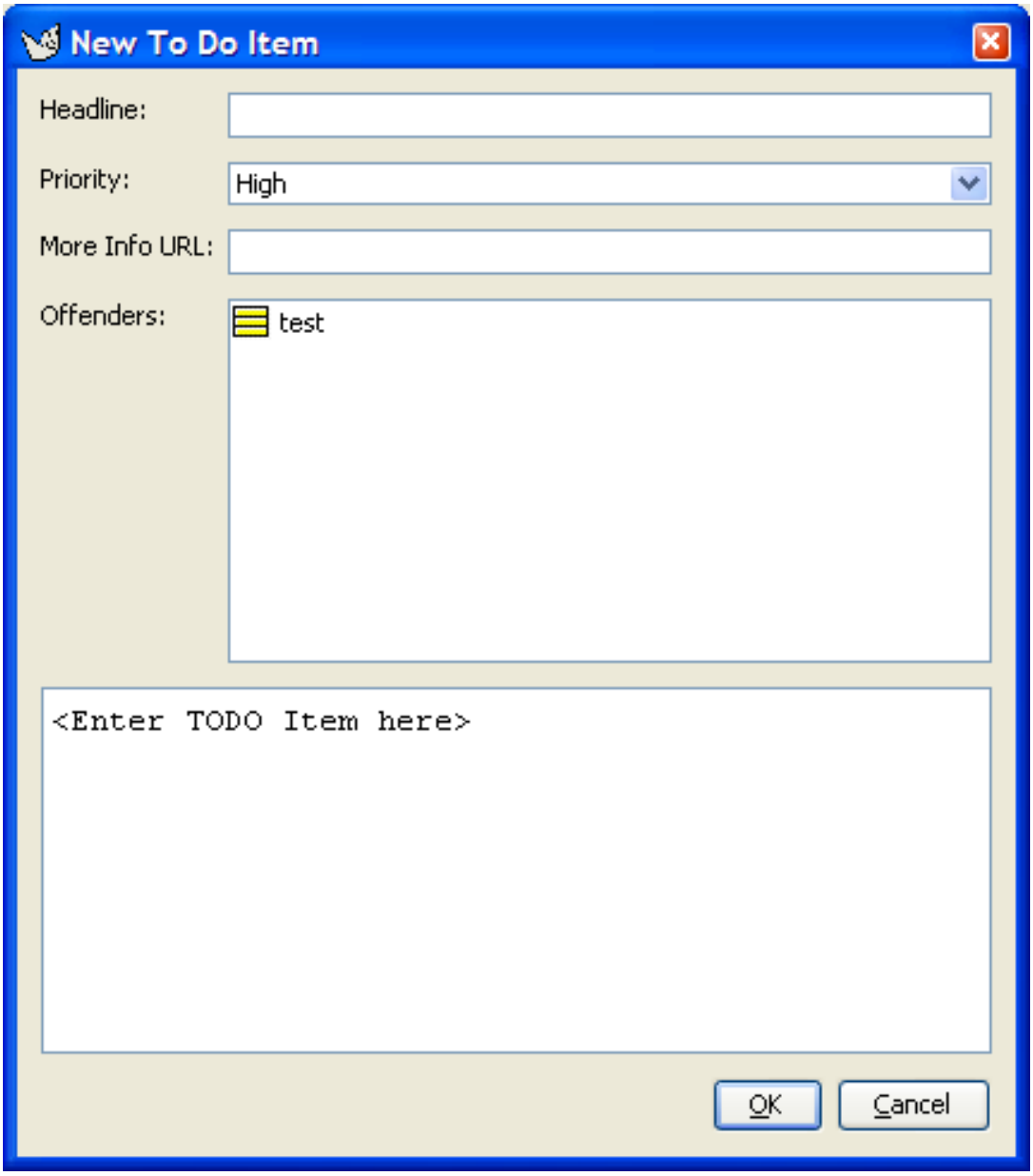
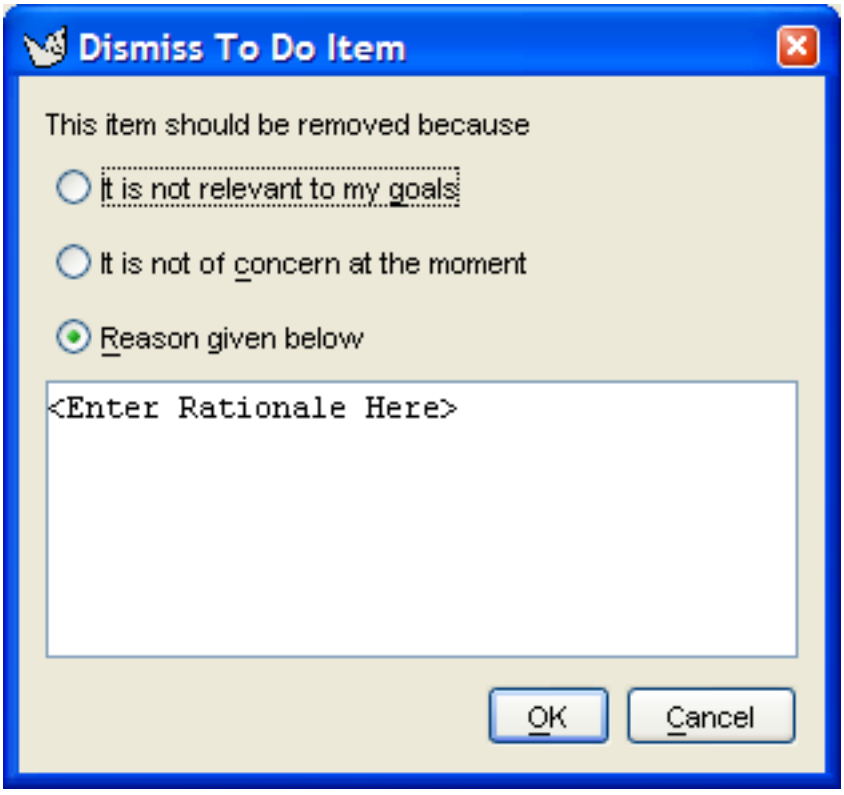

New To Do Item
Headline:
Priority:
High
More Info URL:
Offenders:
test
<Enter TODO Item here>
OK
Cancel

Figura 13.4.



Sugerencia

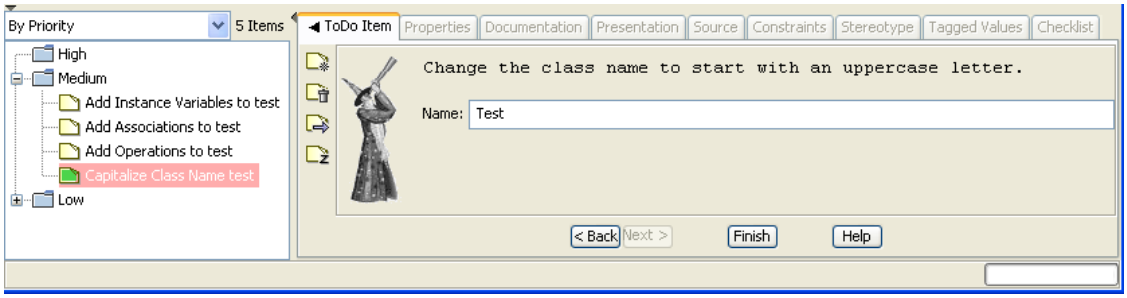
Aviso

.

Sugerencia

13.2.1.

Figura 13.5.



-
-
-

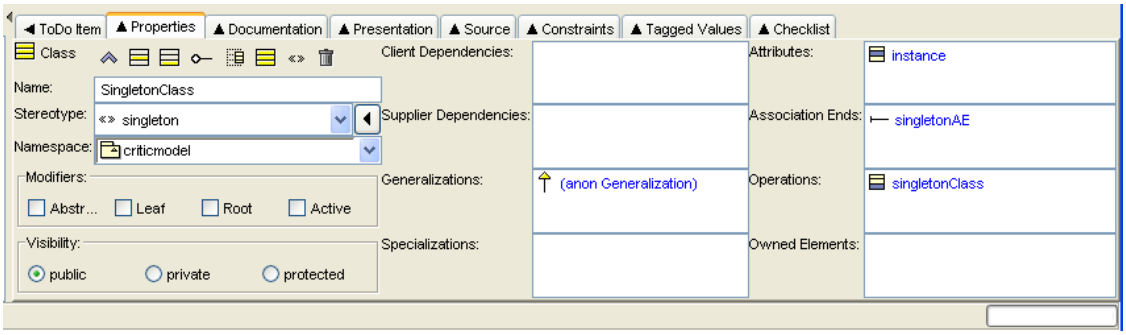
Nota

13.2.2.

13.3.

1.
2.

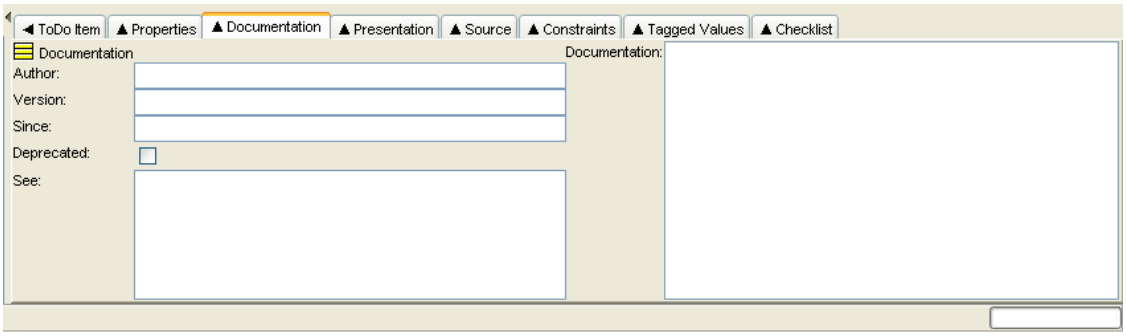
Figura 13.6.



Atención

13.4.

Figura 13.7.



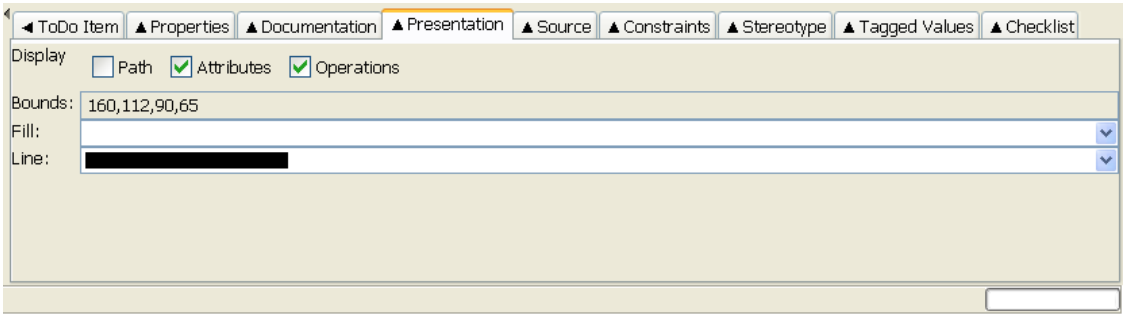
-

-
-
-
-
-
-
-

Sugerencia

13.5.

Figura 13.8.



-
-
-
-
-
-
-
-
-

Figura 13.9.

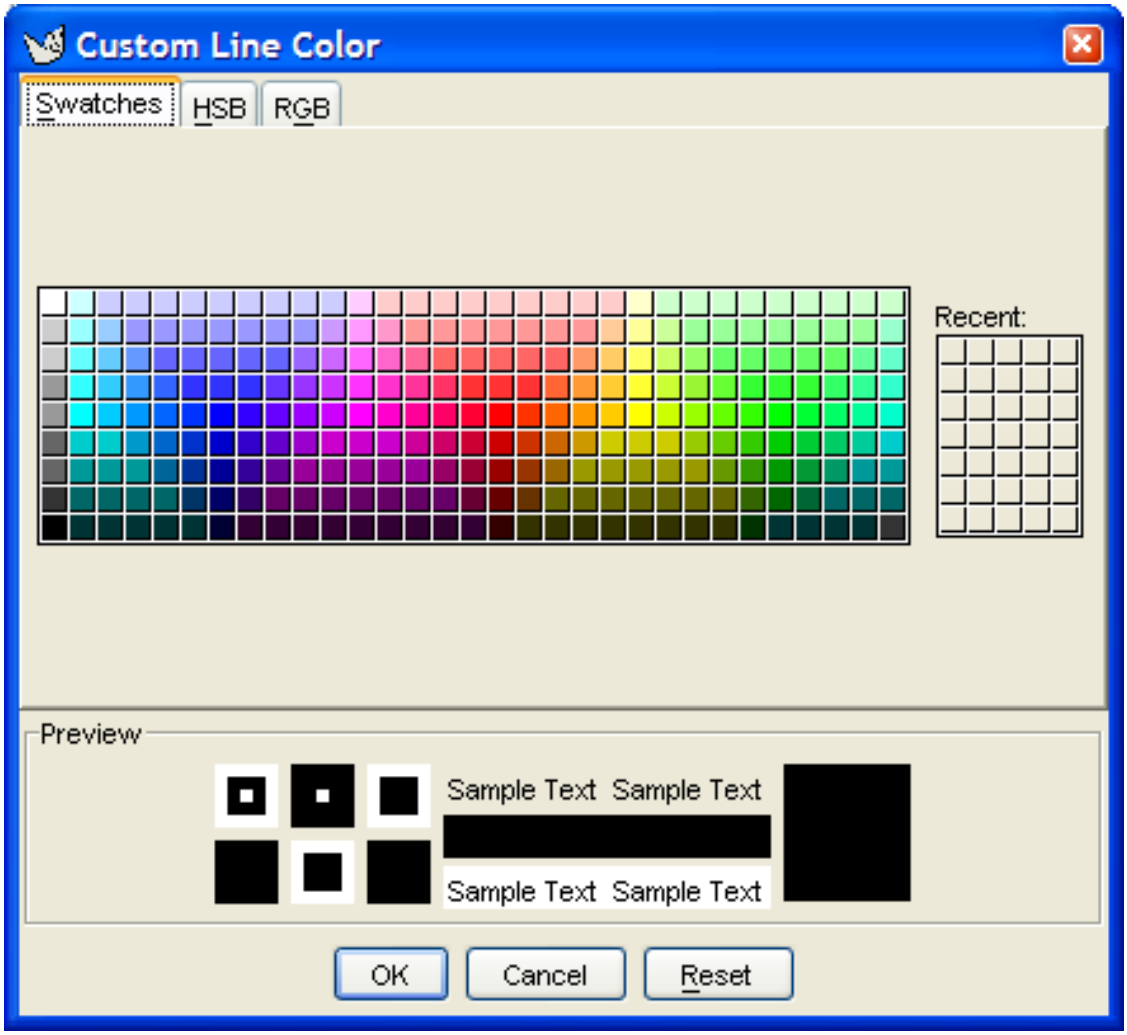

Custom Line Color
Swatches
HSB
RGB
Recent:
Preview
Sample Text Sample Text
Sample Text Sample Text
OK
Cancel
Reset

Figura 13.10.

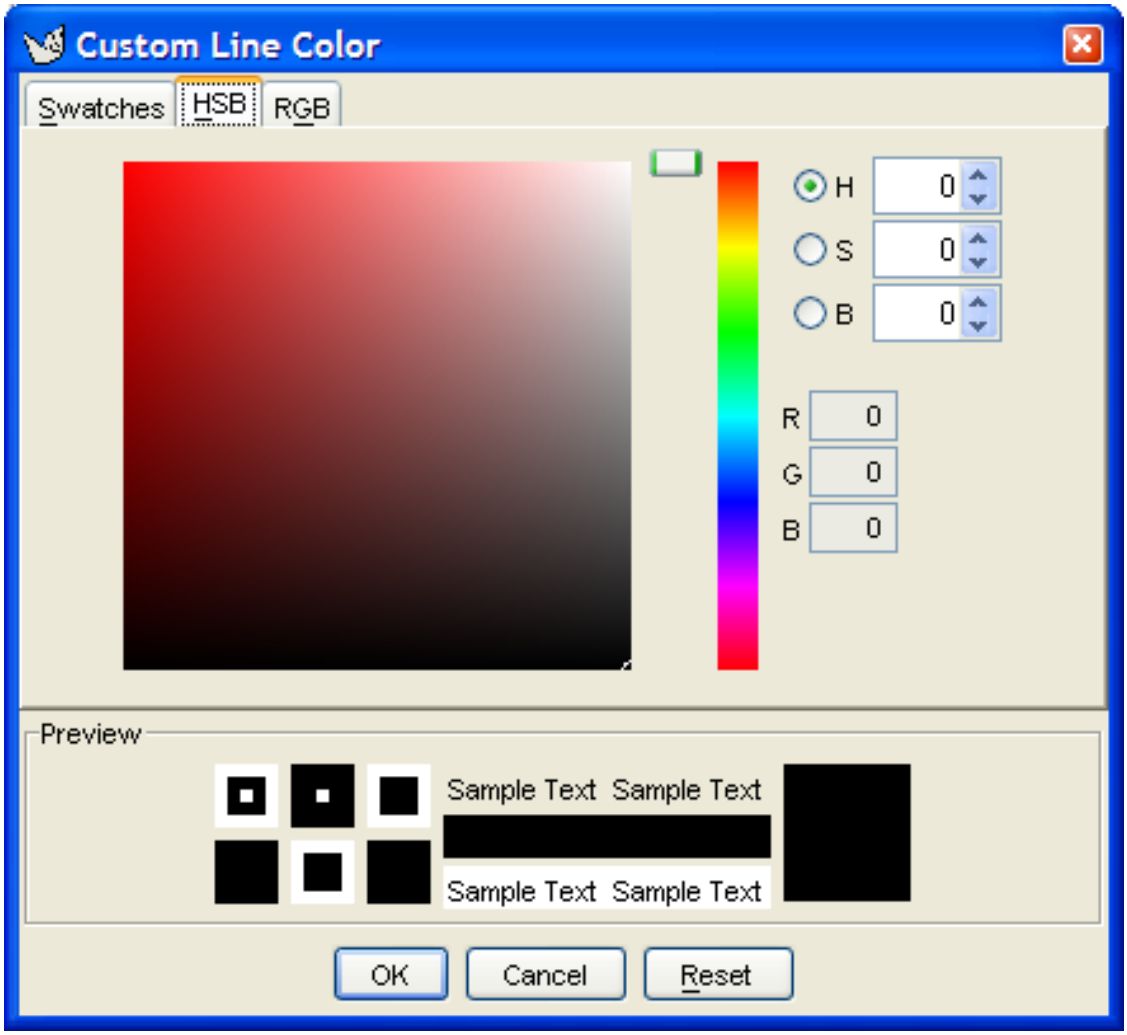
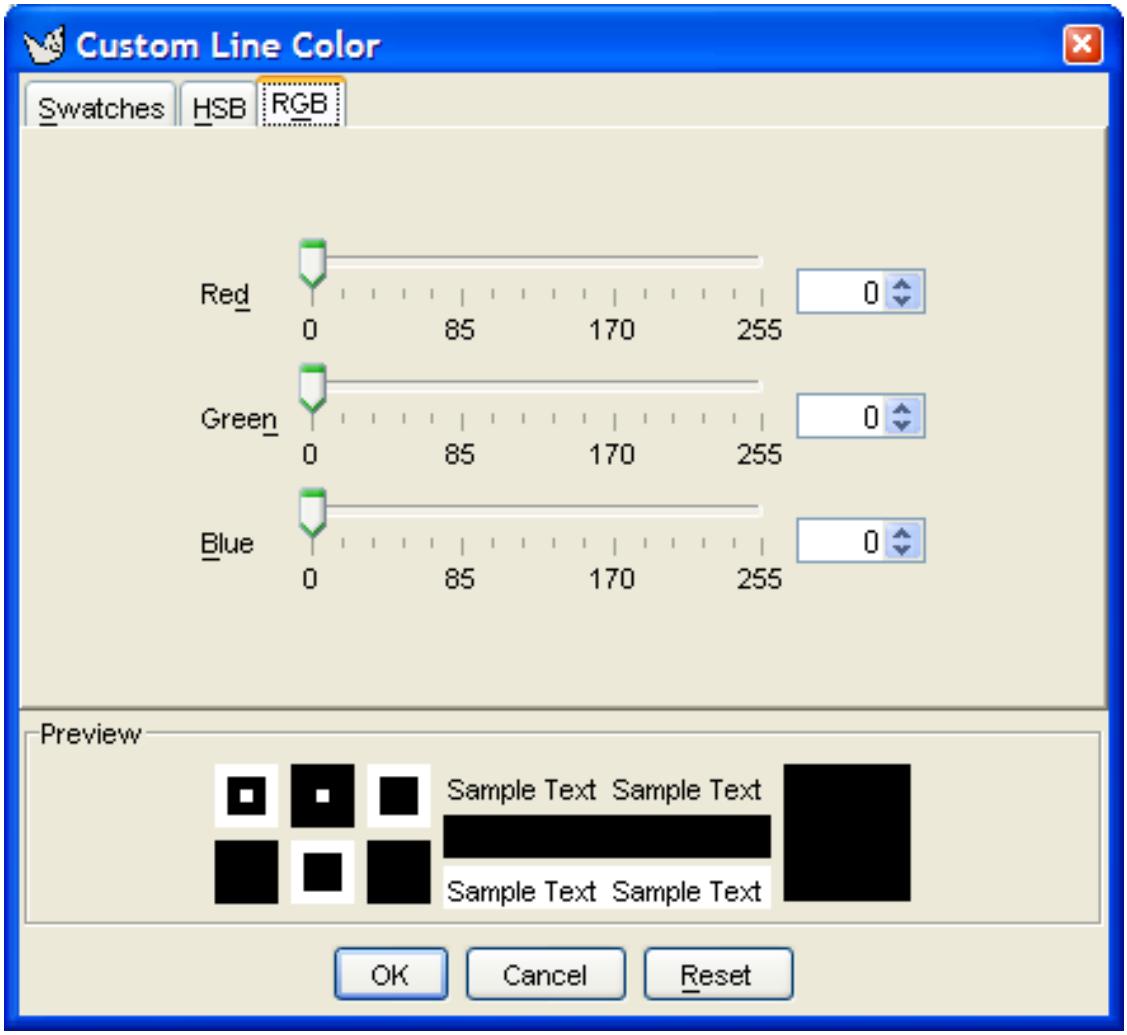

Custom Line Color
Swatches
HSB
RGB
H
S
B
0
0
0
R
G
B
0
0
0
Preview
Sample Text Sample Text
Sample Text Sample Text
OK
Cancel
Reset

Figura 13.11.



13.6.

Figura 13.12.

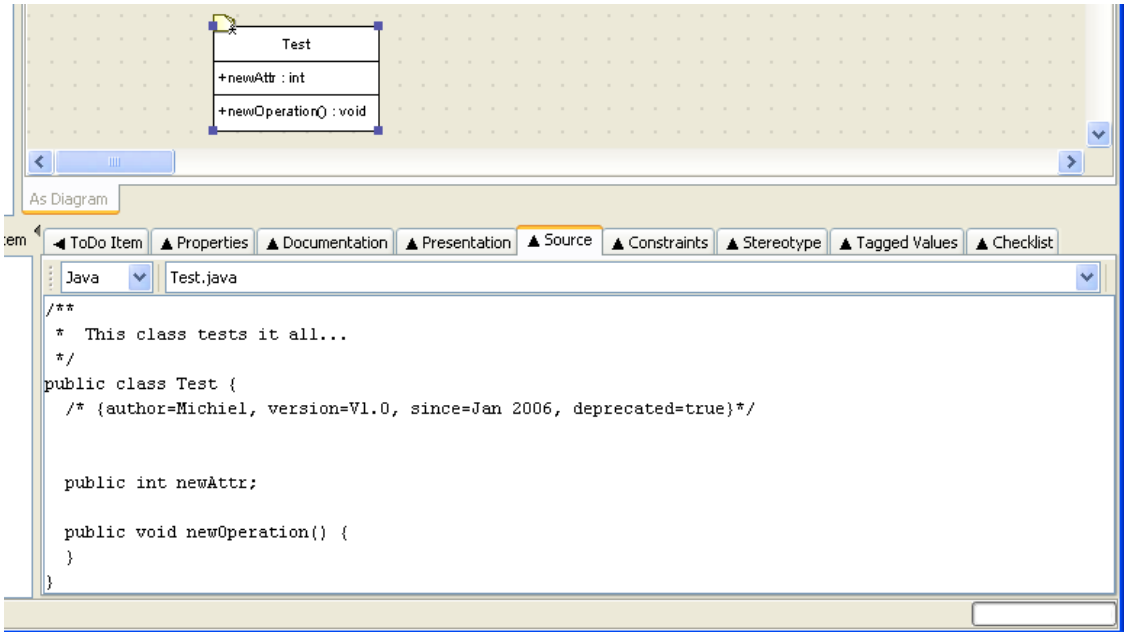
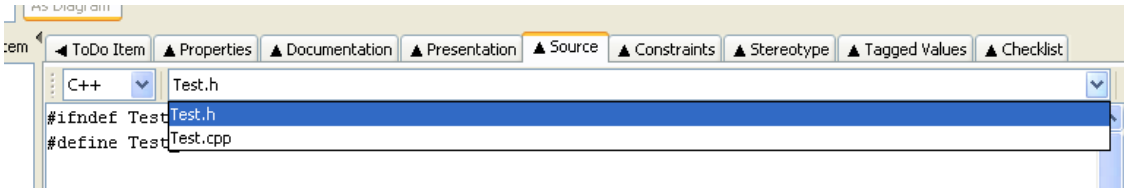


Figura 13.13.

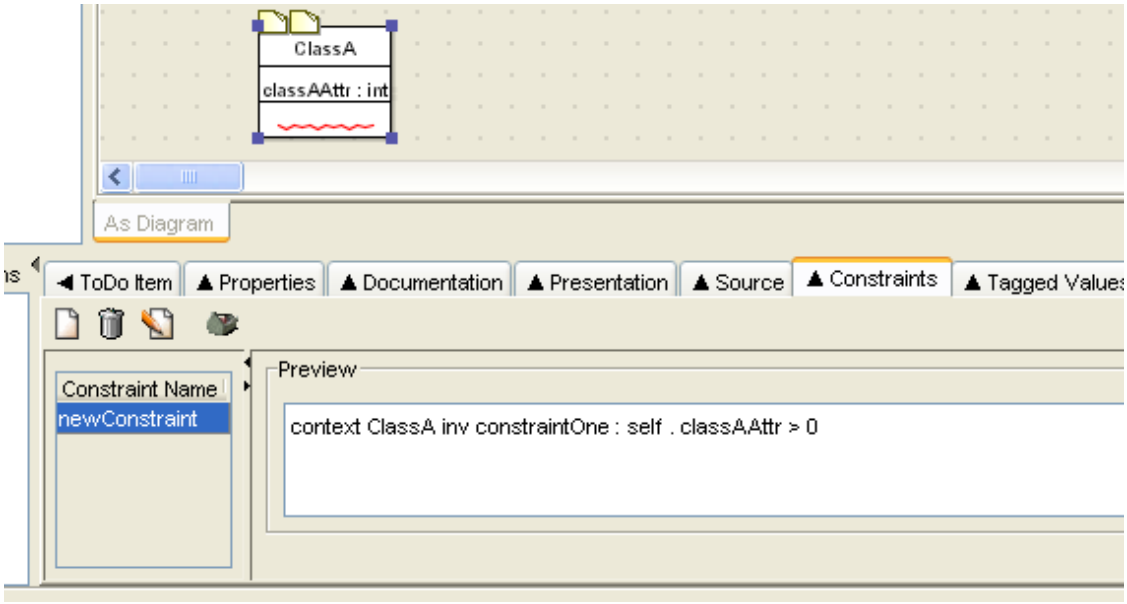


13.7.

Atención

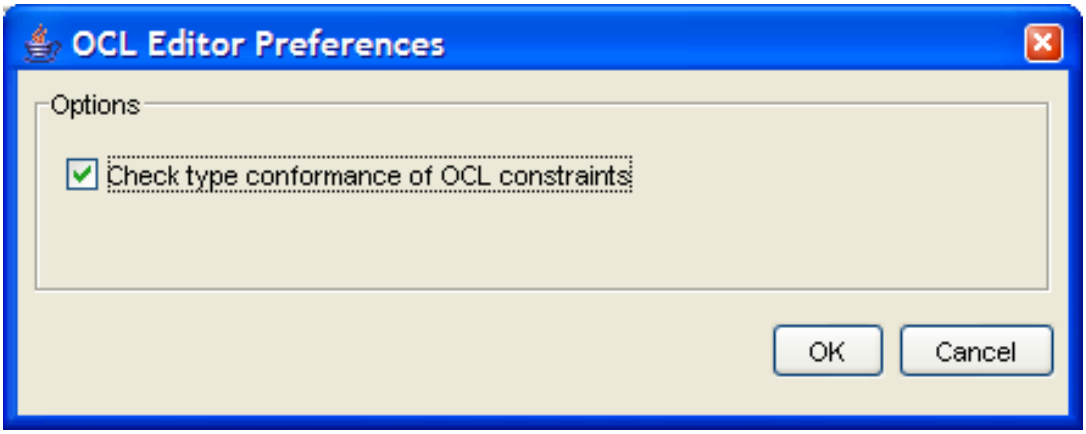
-
-
-
-
-
-

Figura 13.14.



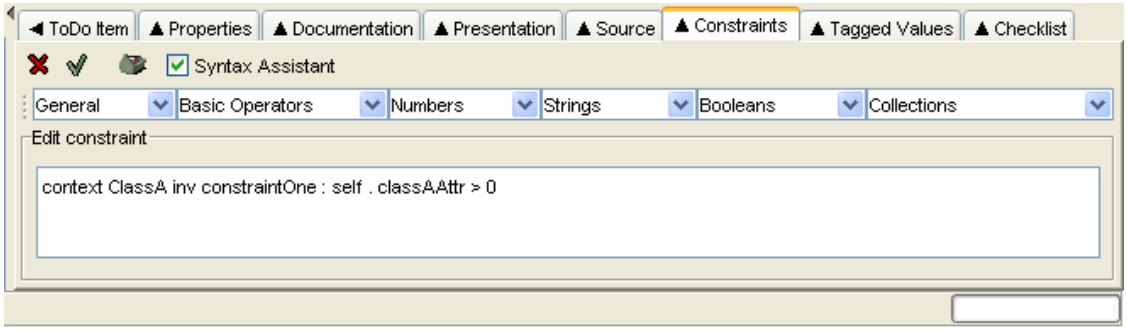
- **Aviso**
- **Atención**
- **Atención**

Figura 13.15.



13.7.1.

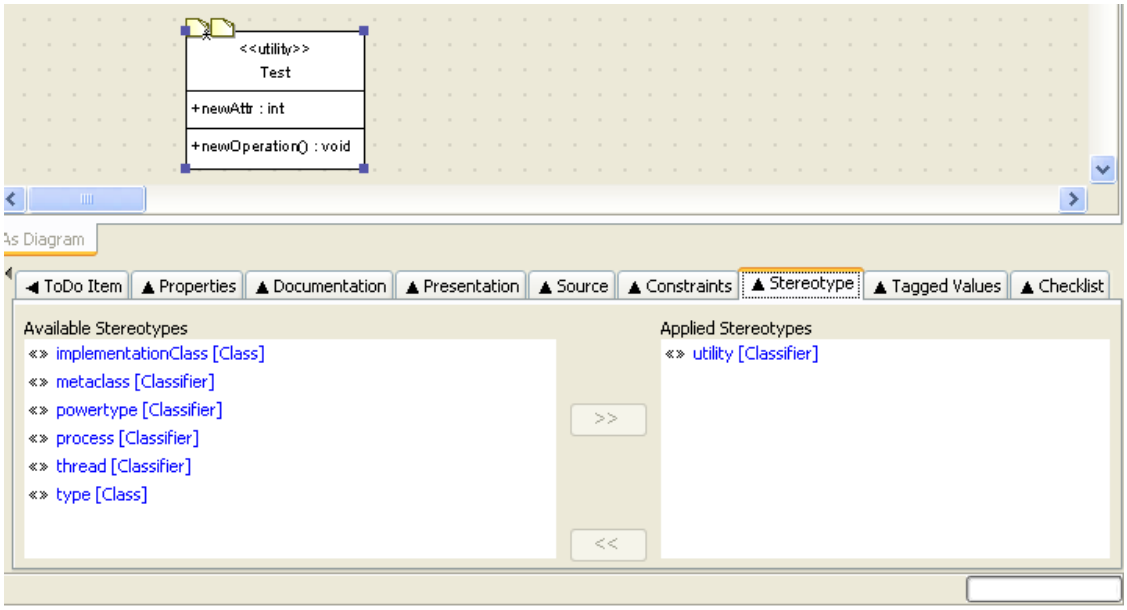
Figura 13.16.



Aviso

13.8.

Figura 13.17.

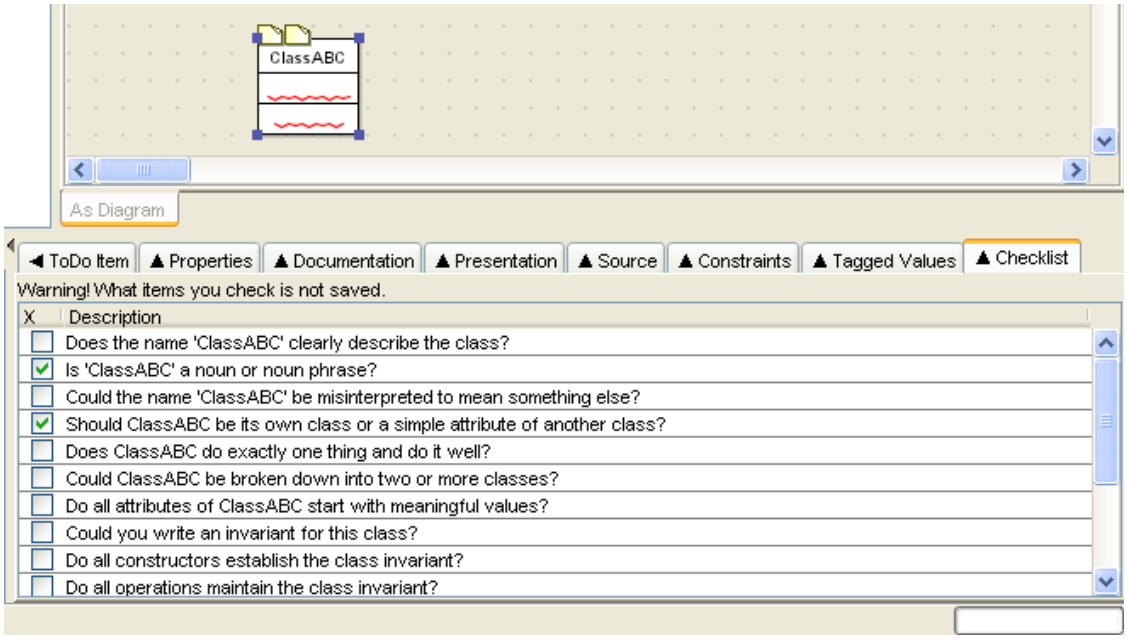


13.9.

Nota

13.10.

Figura 13.18.

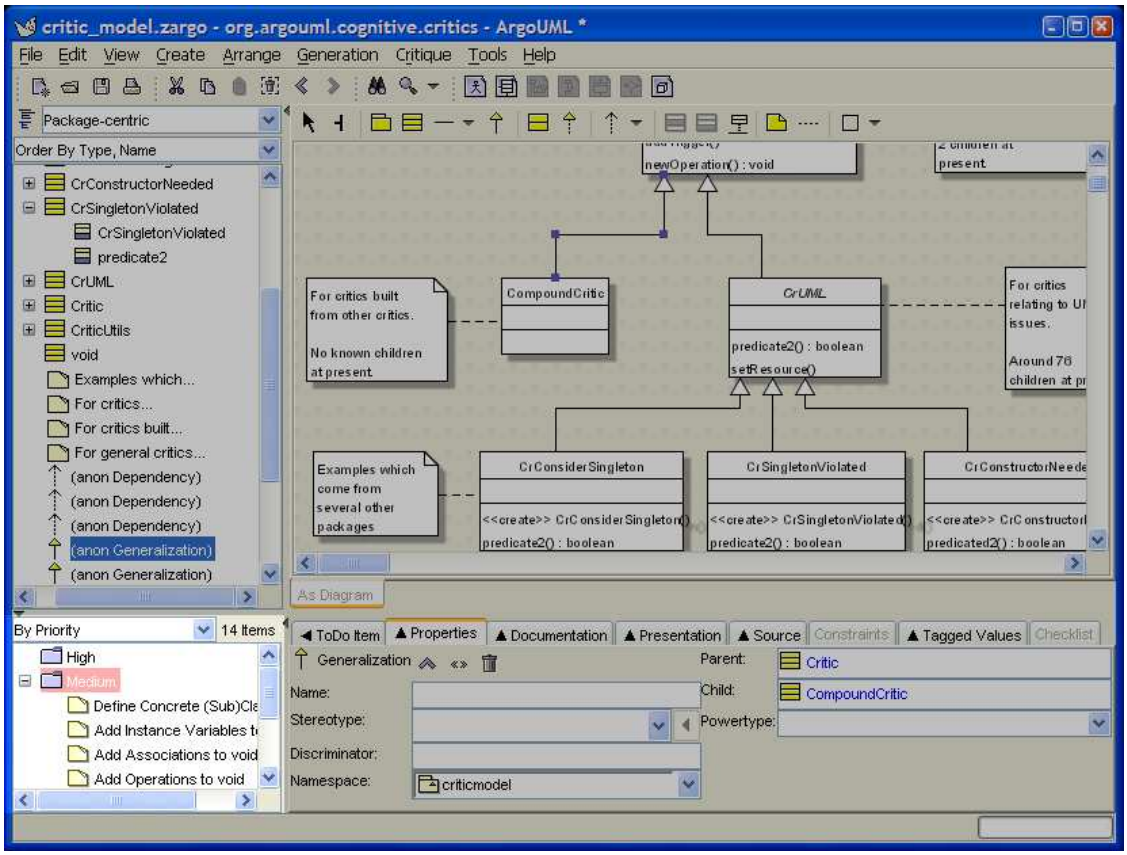


Atención

Capítulo 14.

14.1.

Figura 14.1.



14.2.

14.2.1.

14.2.2.

14.2.3.

14.2.4.

14.3.

-
-
-

Atención

-
-
-

14.4.

Capítulo 15.

15.1.

Nota

15.1.1.

15.1.2.

15.2.

15.3.

15.3.1.

:
:

15.3.2.

Atención

15.3.3.

15.4.

15.4.1.

15.4.2.

Nota

15.4.3.

1.
2.

Nota

15.4.4.

15.4.5.

15.4.6.

15.4.7.

15.4.8.

15.4.9.

15.4.10.

15.4.11.

15.4.12.

15.4.13.

Atención

15.4.14.

15.4.15.

15.4.16.

Atención

15.4.17.

15.4.18.

Nota

15.4.19.

15.5.

15.5.1.

15.5.2.

15.5.3.

Atención

15.5.4.

Atención

15.6.

Nota

15.6.1.

15.6.2.

Atención

15.6.3.

15.7.

15.7.1.

Atención

15.7.2.

Atención

15.7.3.

15.7.4.

15.7.5.

15.7.6.

15.7.7.

15.7.8.

15.7.9.

15.7.10.

15.7.11.

15.7.12.

Atención

15.7.13.

15.7.14.

15.7.15.

15.8.

15.8.1.

1.
2.
3.

15.8.2.

15.8.3.

15.8.4.

15.8.5.

15.8.6.

15.8.7.

15.8.8.

15.8.9.

15.8.10.

15.8.11.

15.8.12.

15.8.13.

15.8.14.

Aviso

15.8.15.

15.8.16.

15.9.

15.9.1.

Aviso

15.9.2.

15.9.3.

15.9.4.

15.9.5.

15.9.6.

Atención

15.9.7.

15.10.

15.11.

15.11.1.

Atención

15.11.2.

Nota

15.12.

15.12.1.

15.13.

15.13.1.

15.13.2.

15.13.3.

15.13.4.

Atención

15.14.

15.14.1.

15.15.

15.16.

15.16.1.

15.16.2.

Atención

15.16.3.

15.16.4.

15.16.5.

Atención

15.16.6.

15.16.7.

15.16.8.

15.16.9.

15.16.10.

15.17.

15.17.1.

15.17.2.

15.17.3.

15.17.4.

15.17.5.

Parte 3.

Capítulo 16.

16.1.

16.2.

Nota

16.2.1.

.

16.2.2.

Sugerencia

16.2.3.

Nota

.
.

Nota

Nota

16.3.

Nota

16.3.1.

- **Sugerencia**
- **Sugerencia**

16.3.2.

Sugerencia

16.3.3.

Nota

- **Nota**
- **Sugerencia**

Atención

Atención

.
.

Atención

16.4.

16.4.1.

.
.
.
.
.

16.4.2.

16.4.3.

Nota

-
-
-
-
-
-

Atención

-
-

16.5.

16.6.

16.6.1.

Aviso

-

Nota

-
-
-
-
-

Atención

16.6.2.

16.6.3.

Nota

:
:

Atención

Aviso

16.7.

16.8.

:
:
:
:
:
:
:
.

Nota

Atención

Aviso

16.8.1.

16.8.2.

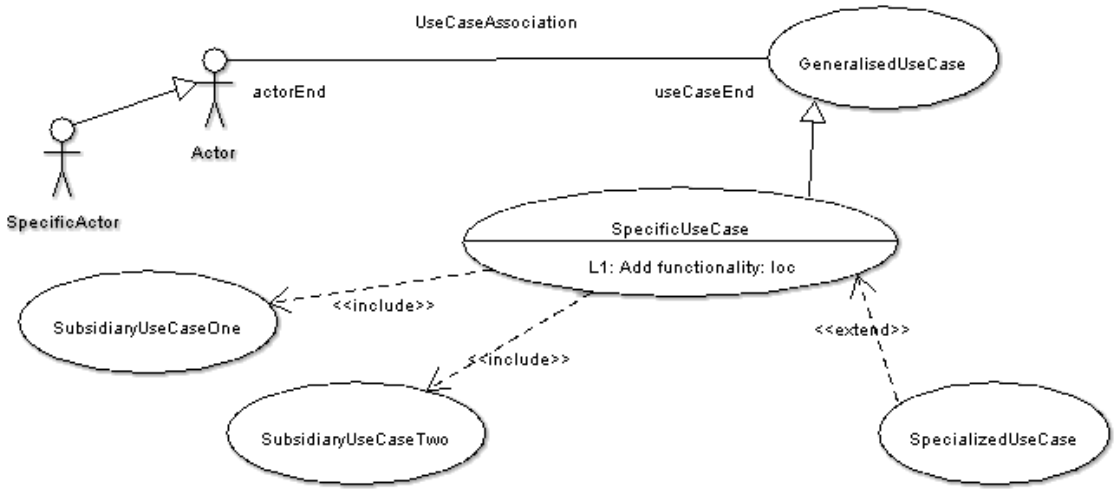
16.8.3.

Sugerencia

Capítulo 17.

17.1.

Figura 17.1.



17.1.1.

Nota

17.2.

17.2.1.

- **Sugerencia**

- **Nota**

17.2.2.

Sugerencia

Sugerencia

Aviso

17.2.3.

Nota

- 

  Atención

-

17.3.

Atención

17.3.1.

- 

  Sugerencia

-
-

Nota

17.3.2.

Sugerencia

Aviso

17.3.3.

Nota

.
.

.
.

.
.

Nota

.
.
.

17.4.

17.4.1.

.

Nota

17.4.2.

17.4.3.

Sugerencia

Nota

Sugerencia

17.5.

17.6.

17.7.

Atención

17.8.

Nota

17.8.1.

Nota

.

Nota

17.8.2.

Aviso

17.8.3.

Sugerencia

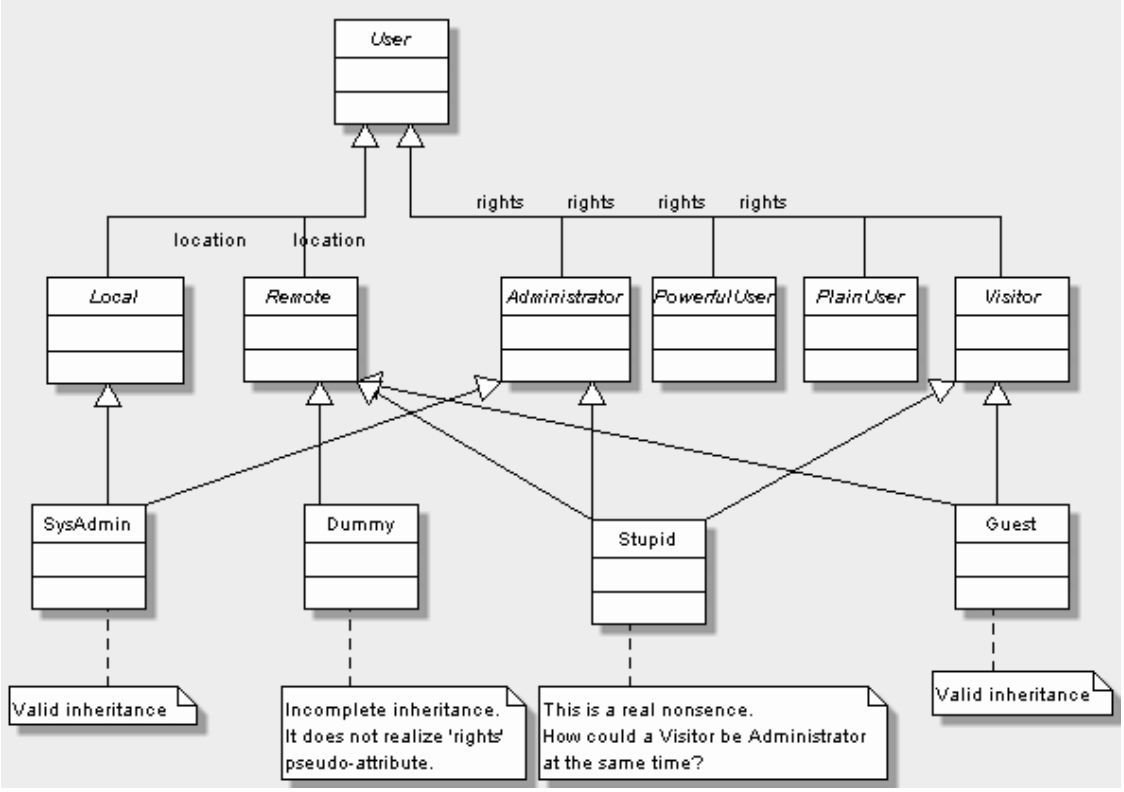
Nota

Nota

Sugerencia

Sugerencia

Figura 17.2.



17.9.

17.9.1.

Nota

Nota

.

Nota

17.9.2.

Sugerencia

Aviso

17.9.3.

Sugerencia

Nota

Nota

- .
- .
- .
- .

17.10.

17.10.1.

Nota

Nota

- .

Nota

17.10.2.

Aviso

17.10.3.

Sugerencia

Nota

Capítulo 18.

18.1.

Figura 18.1.

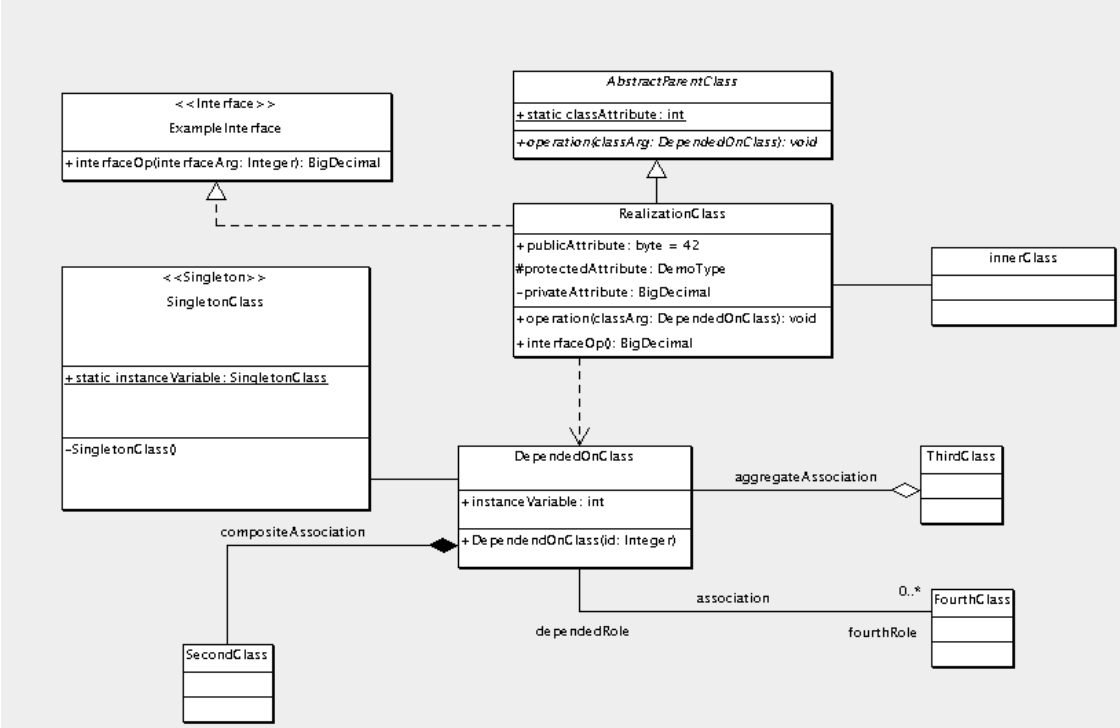


Figura 18.2.

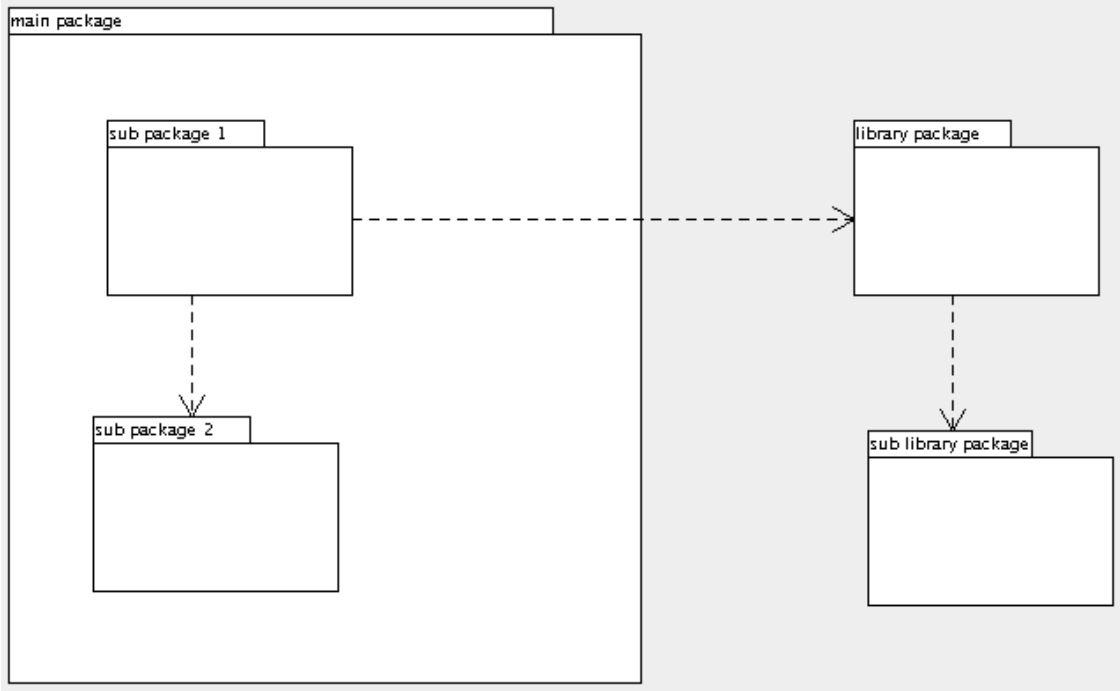


Figura 18.3.

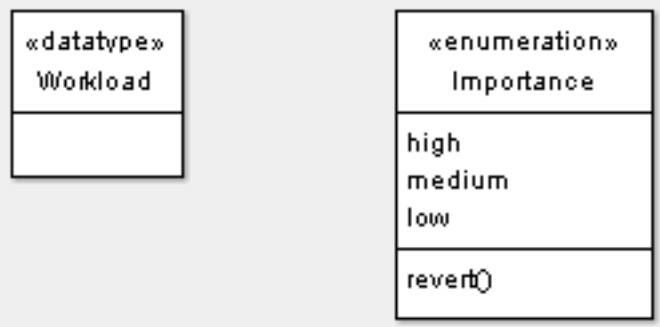
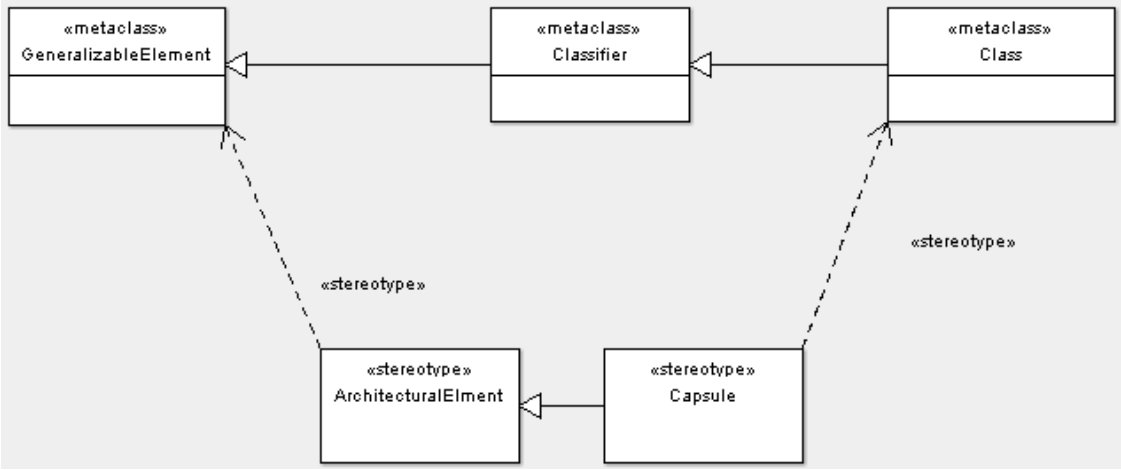


Figura 18.4.



18.1.1.

18.2.

Nota

18.2.1.

Nota

18.2.2.

Aviso

18.2.3.

Nota

.

Sugerencia

:

.

Sugerencia

:

.

18.3.

18.4.

18.5.

18.6.

18.6.1.

.

:

.

Nota

Nota

18.6.2.

Nota

Aviso

18.6.3.

Nota

.

Atención

.
.

.
.

.
.

Sugerencia

18.7.

18.7.1.

- `transient`.
-

Nota

18.7.2.

Sugerencia

Aviso

18.7.3.

Nota

Nota

.
.
.
.

.
.
.

Nota

Atención

18.8.

18.8.1.

:
:

Nota

Nota

18.8.2.

Sugerencia

Aviso

18.8.3.

Nota

Sugerencia

Sugerencia

-
-
-
-
-

Importante

-
-
-

Atención

-
-

Atención

-
-
-
-

Atención

18.9.

18.9.1.

-

Atención

Nota

18.9.2.

Sugerencia

Aviso

18.9.3.

Nota

Nota

Atención

.

.

Nota

.

Nota

Sugerencia

.

18.10.

Sugerencia

18.10.1.

-
-
-

Nota

Nota

18.10.2.

Atención

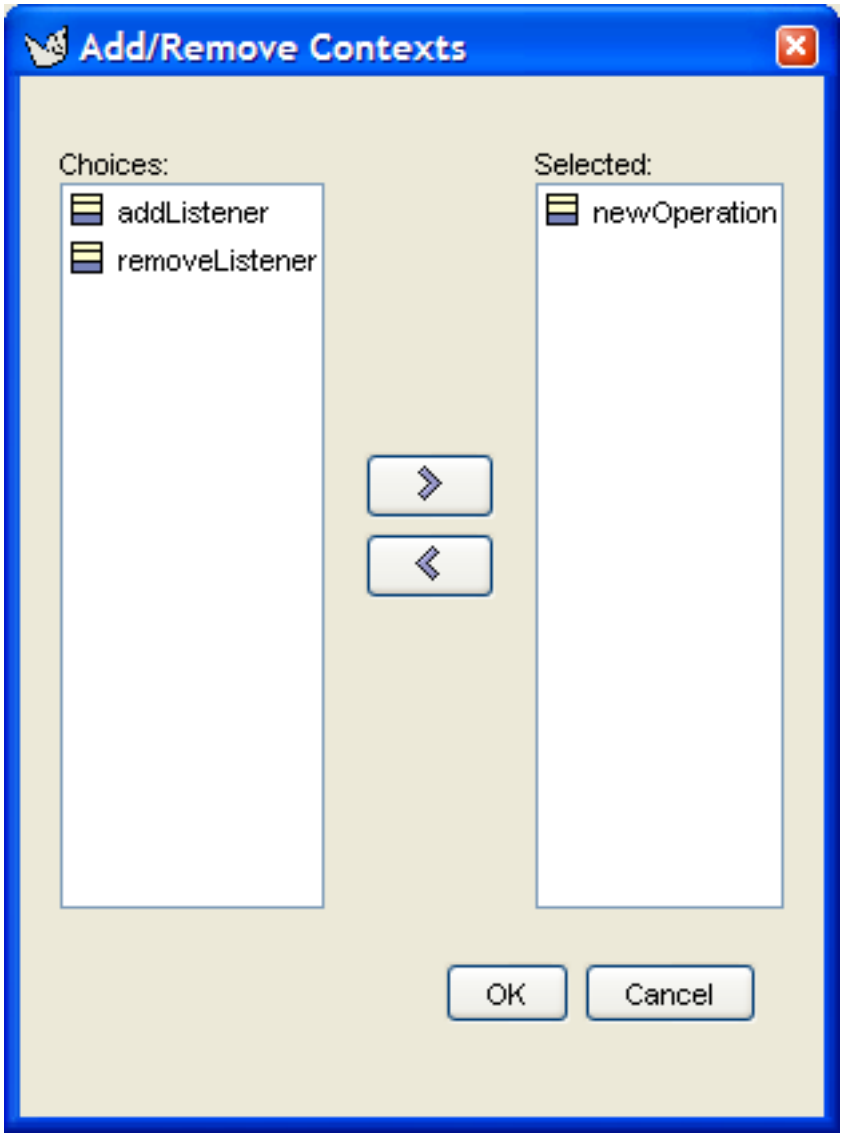
Aviso

18.10.3.

Nota

-

Figura 18.5.



18.11.

18.12.

Nota

18.12.1.

18.12.2.

Nota

:
:

Nota

Nota

18.12.3.

Aviso

18.12.4.

Nota

Sugerencia

Nota

18.13.

Sugerencia

Nota

18.13.1.

- **Sugerencia**

Nota

18.13.2.

Aviso

Nota

18.13.3.

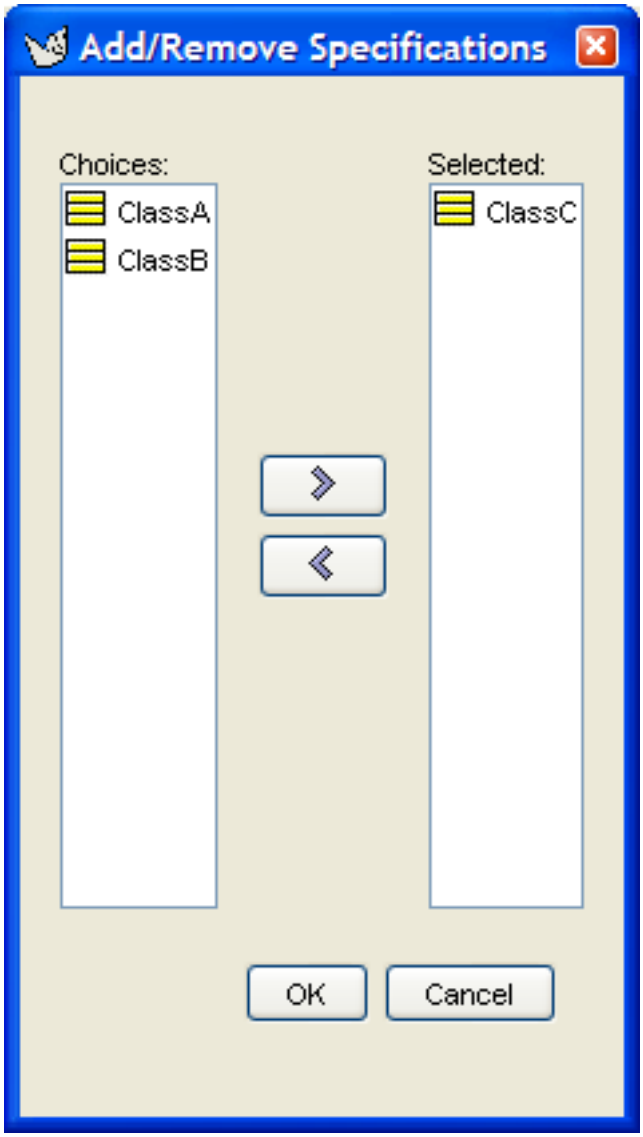
Nota

Sugerencia

- **Nota**

:

Figura 18.6.



Nota

18.14.

18.14.1.

Nota

.

Nota

18.14.2.

Aviso

18.14.3.

Sugerencia

Nota

Nota

18.15.

Nota

18.16.

18.16.1.

- **Aviso**

- **Nota**

Nota

18.16.2.

Aviso

18.16.3.

Nota

- **Atención**

Nota

Atención

18.17.

Atención

18.17.1.

Nota

.

Nota

Nota

18.17.2.

Aviso

18.17.3.

Atención

Nota

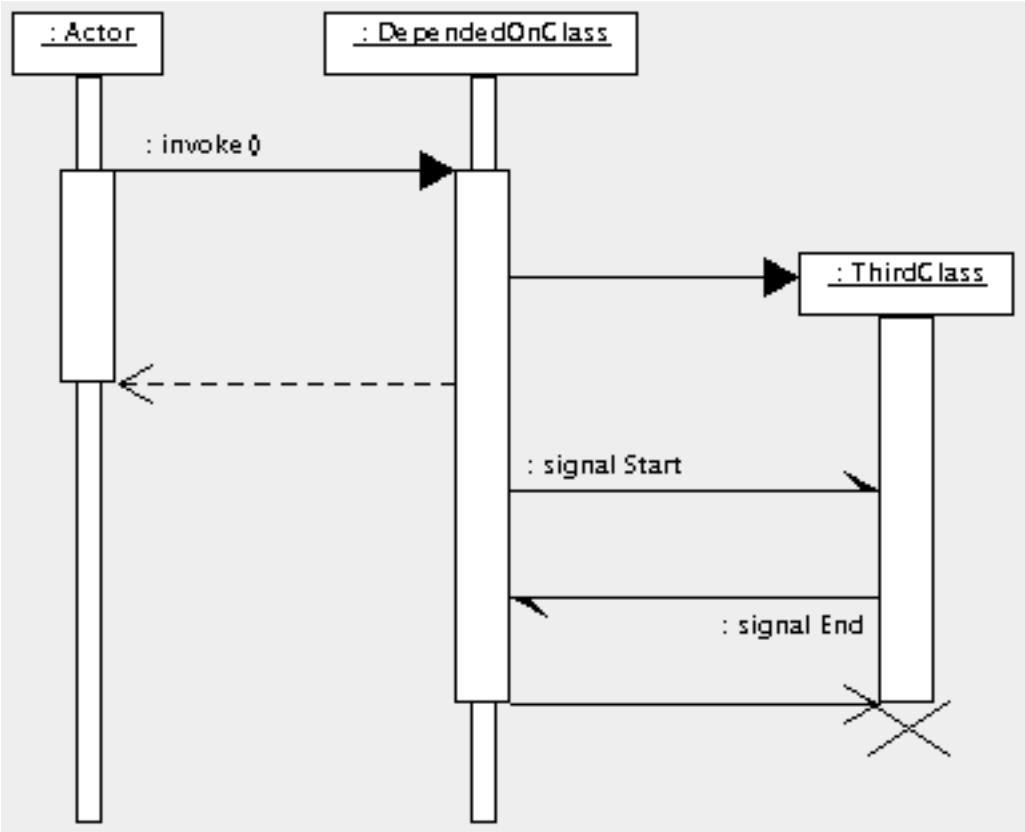
Nota

Capítulo 19.

19.1.

Atención

Figura 19.1.



19.1.1.

19.2.

Atención

19.2.1.

Atención

:
:

Nota

Nota

19.2.2.

Aviso

19.2.3.

Nota

Atención

Nota

19.3.

:
:
:
:
:

Nota

Atención

19.3.1.

Sugerencia

Atención

Atención

.

Nota

Nota

19.3.2.

Aviso

19.3.3.

Sugerencia

Atención

Atención

.

.

Aviso

19.4.

Nota

19.5.

Nota

19.6.

Nota

19.7.

Nota

19.8.

Nota

19.9.

19.9.1.

Nota

Atención

:
:

Nota

Nota

19.9.2.

Aviso

19.9.3.

Nota

Capítulo 20.

20.1.

Figura 20.1.

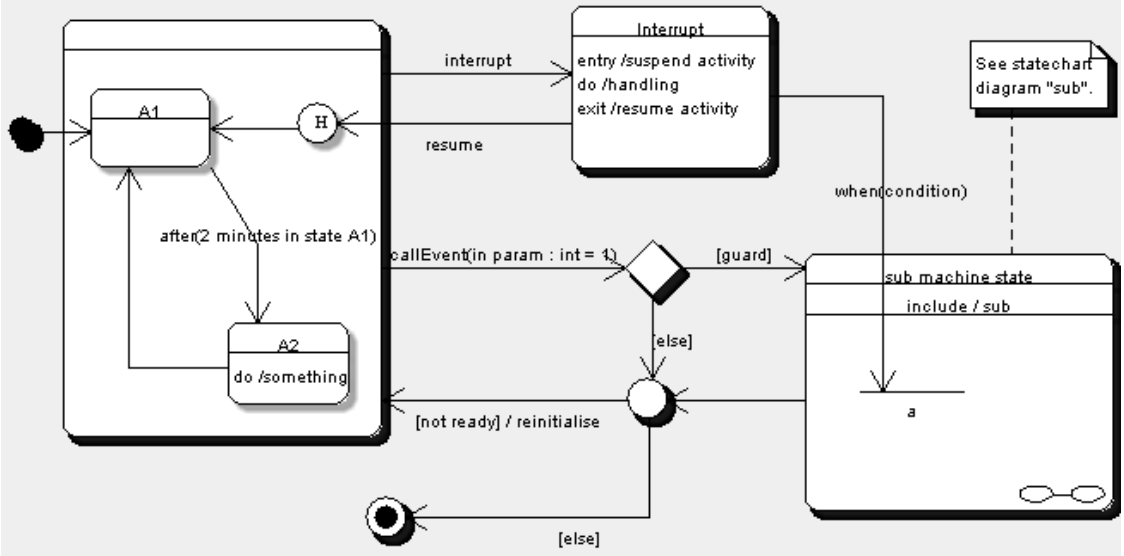
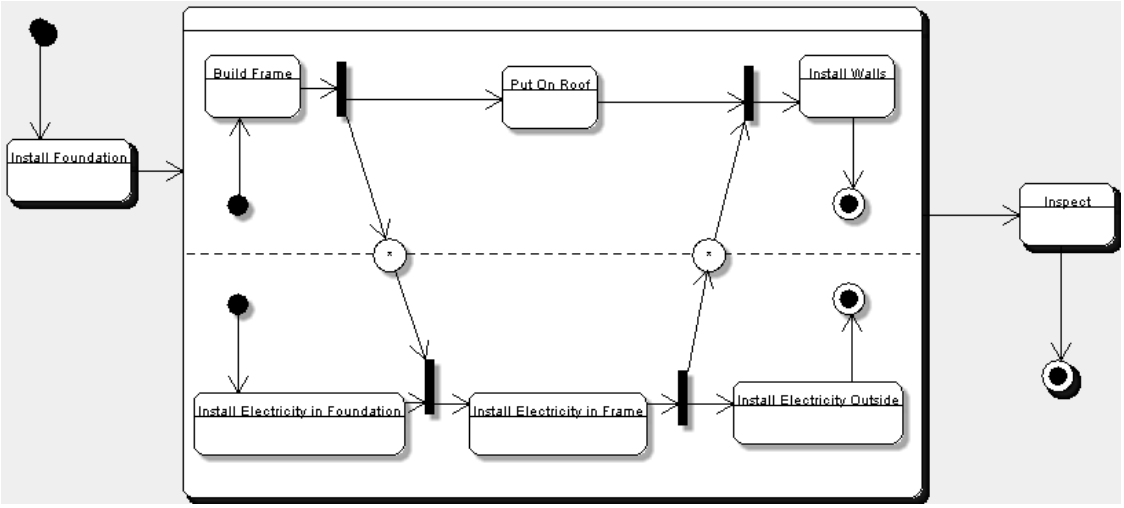


Figura 20.2.



20.1.1.

20.2.

20.2.1.

20.2.2.

Nota

20.2.3.

Nota

Nota

.
.

.
.

.
.

.
.
.

Nota

.

20.3.

.
.
.

-
-
-
-

Atención

20.3.1.

20.3.2.

20.3.3.

Nota

-
-
-

20.4.

-

Aviso

-

20.5.

20.6.

20.7.

20.8.

20.8.1.

Nota

20.8.2.

Aviso

20.8.3.

Nota

Nota

.

.

.

20.9.

.

.

.

.

20.9.1.

.

.

20.9.2.

20.9.3.

Nota

Sugerencia

.

.

.

Aviso

20.10.

20.10.1.

20.10.2.

Aviso

20.10.3.

Nota

20.11.

20.11.1.

20.11.2.

Aviso

20.11.3.

Nota

Sugerencia

20.12.

20.13.

Nota

20.13.1.

20.13.2.

Aviso

20.13.3.

Nota

Sugerencia

:

20.14.

20.15.

Nota

20.16.

Atención

Sugerencia

20.17.

Atención

Sugerencia

20.18.

20.19.

20.20.

20.20.1.

20.20.2.

Aviso

20.20.3.

Nota

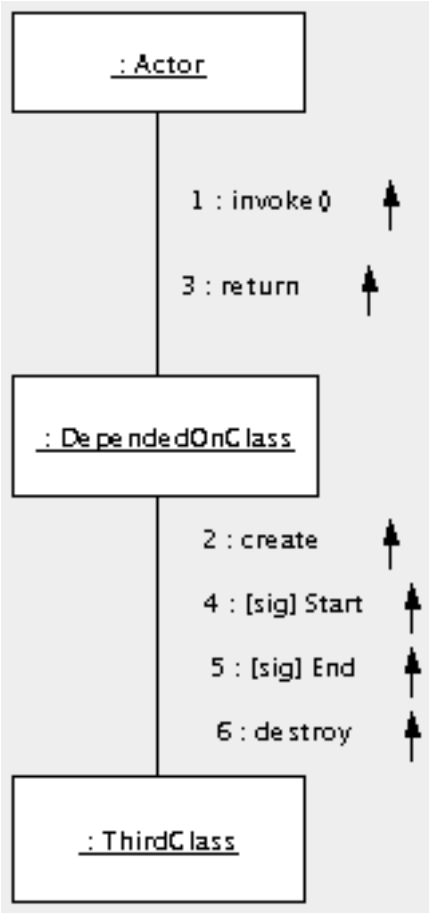
Sugerencia

Capítulo 21.

21.1.

Atención

Figura 21.1.



21.1.1.

21.2.

:

:

Atención

Atención

21.2.1.

Atención

.
.
.

Nota

Nota

21.2.2.

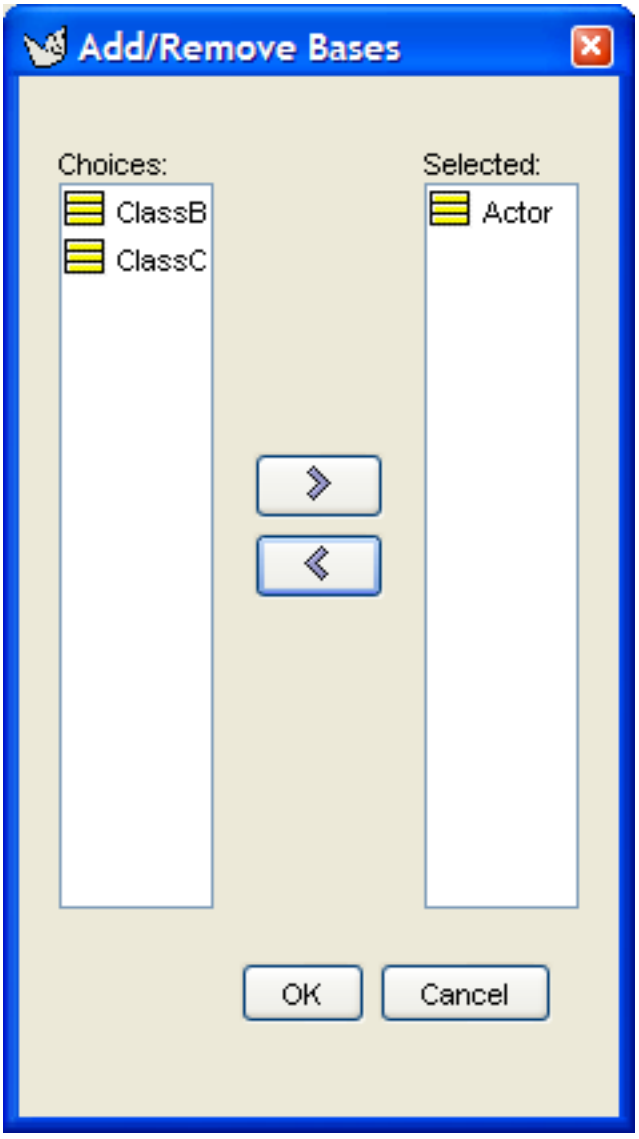
Aviso

21.2.3.

Nota

.

Figura 21.2.


Add/Remove Bases
Choices:
ClassB
ClassC
Selected:
Actor
OK
Cancel

.

21.3.

I / R : C

21.3.1.

Nota

Atención

:

:

Nota

Nota

21.3.2.

Aviso

21.3.3.

Nota

21.4.

Nota

Atención

21.4.1.

-  **Sugerencia**

 **Nota**

21.4.2.

 **Aviso**

 **Nota**

21.4.3.

 **Nota**

21.5.

 **Aviso**

21.5.1.

 **Atención**

Atención

- 

  Nota

Nota

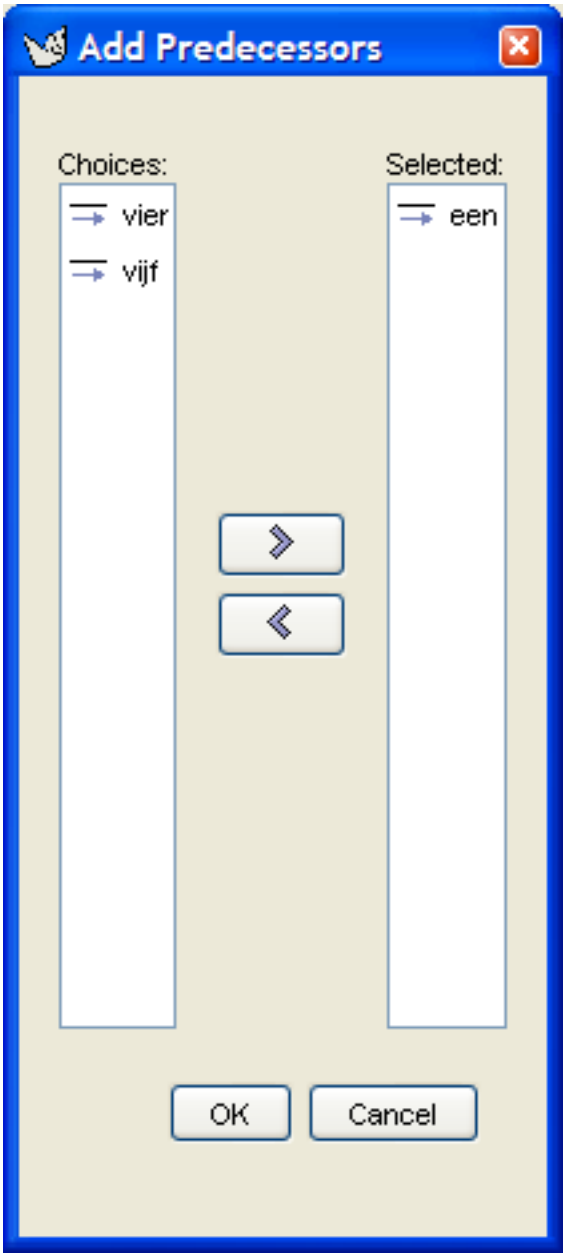
21.5.2.

Aviso

21.5.3.

-
-

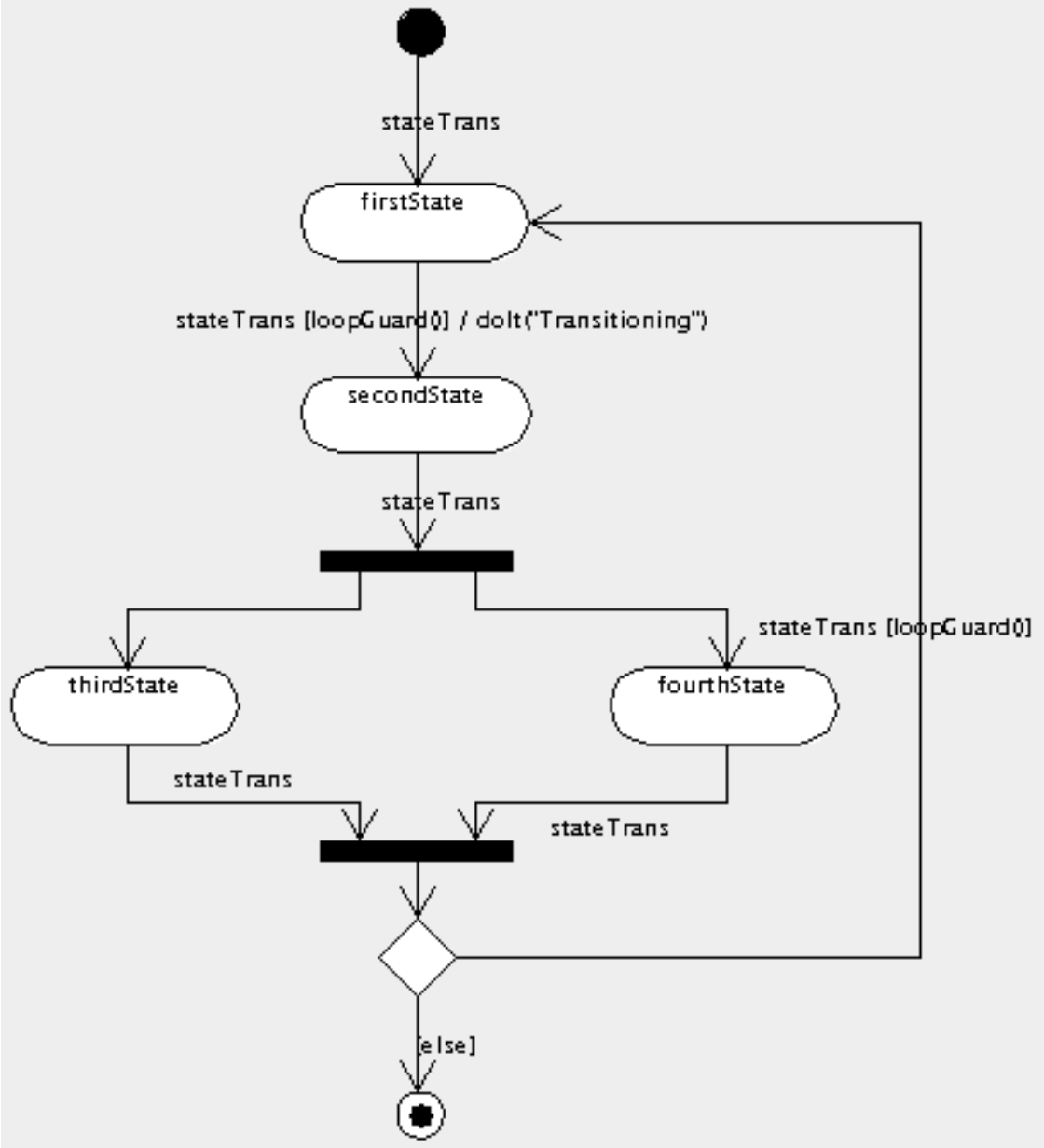
Figura 21.3.


Add Predecessors
Choices:
vier
vijf
Selected:
een
OK
Cancel

Capítulo 22.

22.1.

Figura 22.1.



22.1.1.

22.2.

Atención

Nota

Atención

22.2.1.

22.2.2.

Aviso

22.2.3.

Nota

:
:

22.3.

22.4.

Atención

Nota

22.5.

22.6.

22.7.

22.8.

22.9.

22.10.

22.11.

Capítulo 23.

23.1.

Atención

Figura 23.1.

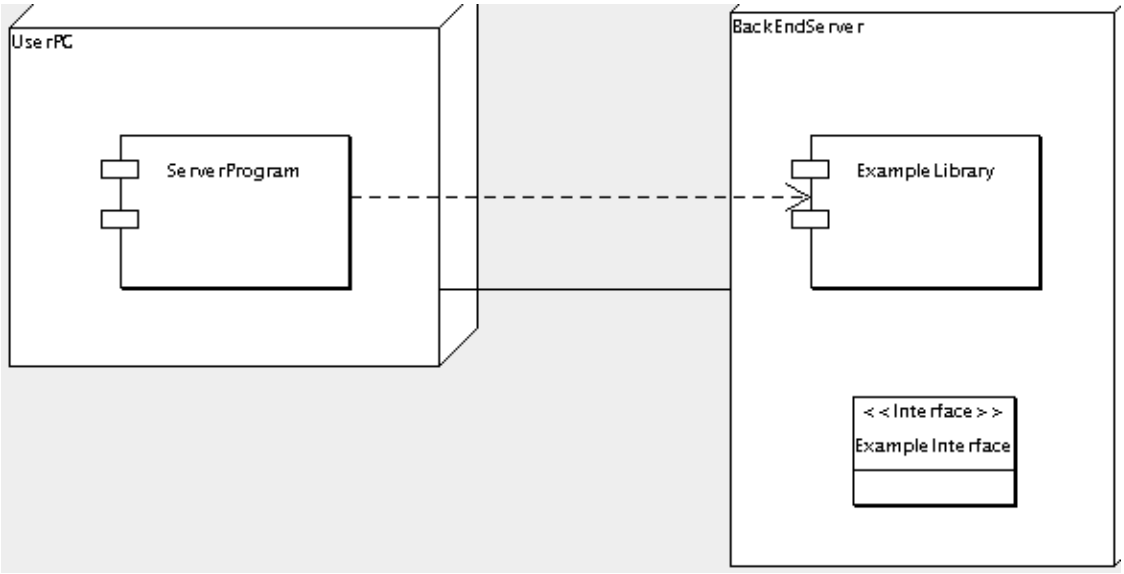
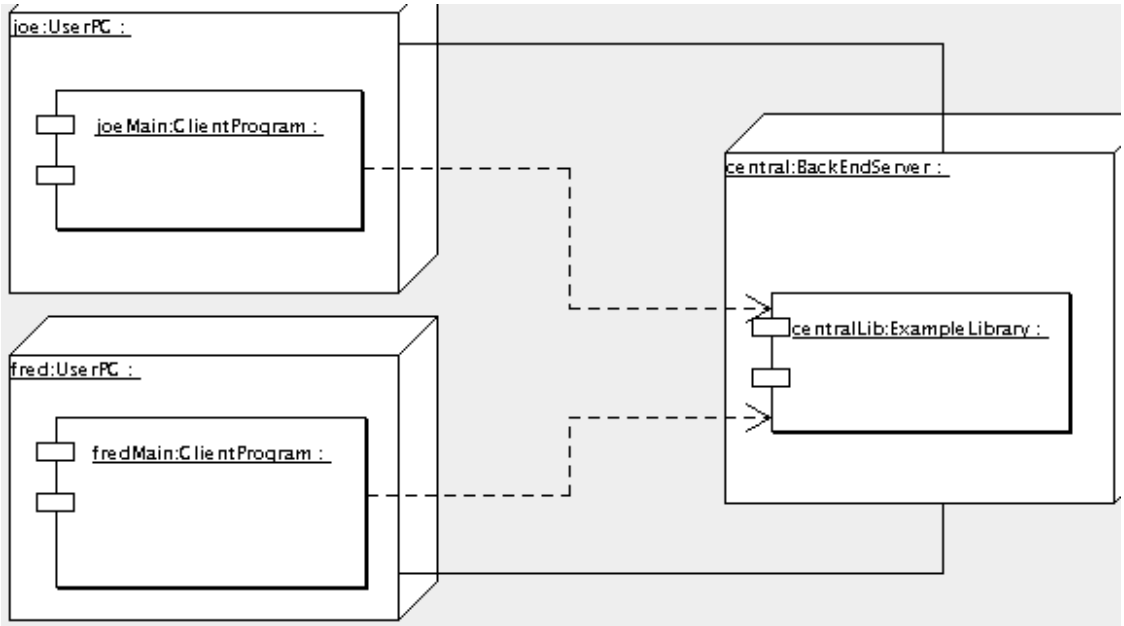


Figura 23.2.



23.1.1.

23.2.

23.2.1.

Aviso

Atención

. . .

Nota

Nota

23.2.2.

Aviso

23.2.3.

Nota

:
:
:

23.3.

Sugerencia

23.3.1.

Aviso

Atención

Nota

23.3.2.

Aviso

23.3.3.

Nota

Atención

23.4.

23.4.1.

Atención

Nota

23.4.2.

Aviso

23.4.3.

Nota

:
:
:

23.5.

Sugerencia

23.5.1.

Atención

Nota

23.5.2.

Aviso

23.5.3.

Nota

Atención

23.6.

Atención

23.7.

Atención

23.8.

Atención

Aviso

23.9.

23.10.

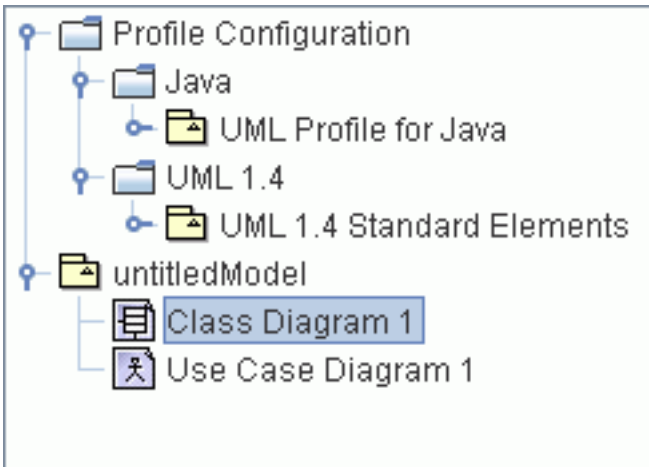
23.11.

Capítulo 24.

Atención

24.1.

Figura 24.1.



24.2.

24.2.1.

- `Integer`
- `String`
- `UnlimitedInteger`

24.2.2.

- `Boolean`

24.2.3.

Tabla 24.1.

access	Permission
appliedProfile	Package
association	AssociationEnd
auxiliary	Class
become	Flow
call	Usage
copy	Flow
create	BehavioralFeature
create	CallEvent
create	Usage
derive	Abstraction
destroy	BehavioralFeature
destroy	CallEvent
document	Abstraction
executable	Abstraction
facade	Package
file	Abstraction
focus	Class
framework	Package
friend	Permission
global	AssociationEnd
implementation	Class
implementation	Generalization
implicit	Association
import	Permission

instantiate	Usage
invariant	Constraint
library	Abstraction
local	AssociationEnd
metaclass	Class
metamodel	Package
modelLibrary	Package
parameter	AssociationEnd
postcondition	Constraint
powertype	Class
precondition	Constraint
process	Classifier
profile	Package
realize	Abstraction
refine	Abstraction
requirement	Comment
responsibility	Comment
self	AssociationEnd
send	Usage
signalflow	ObjectFlowState
source	Abstraction
stateInvariant	Constraint
stub	Package
systemModel	Package
table	Abstraction

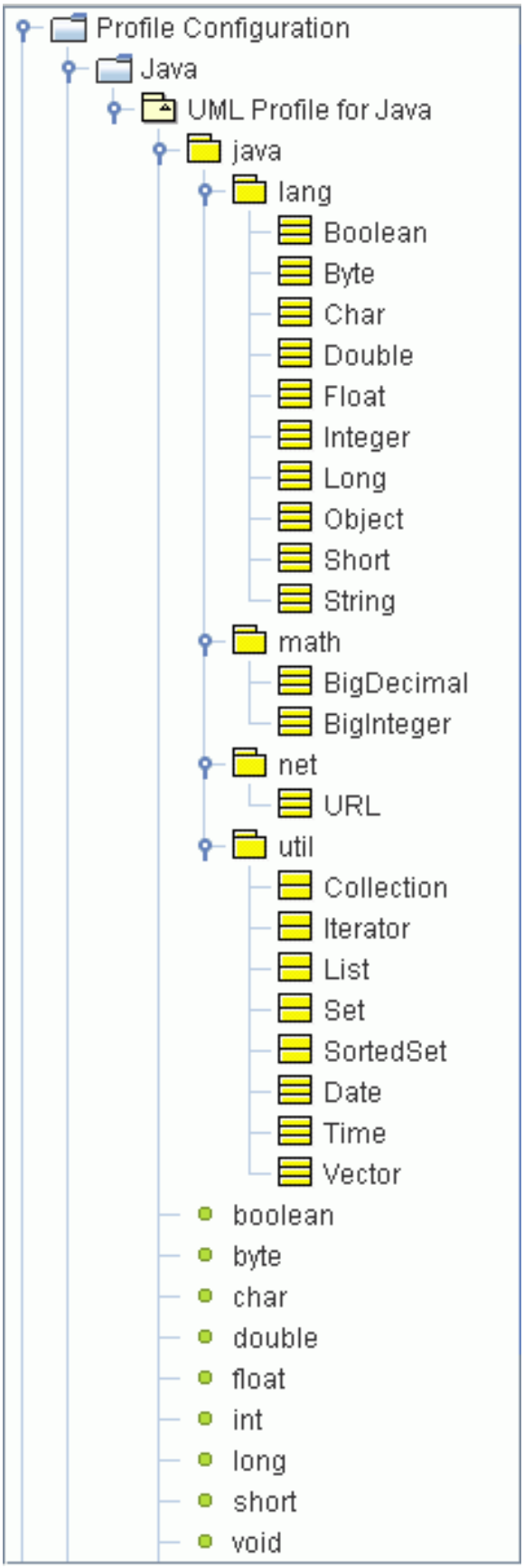
thread	Classifier
topLevel	Package
trace	Abstraction
type	Class

24.2.4.

- `derived`
- `documentation`
- `persistence`
- `persistent`
- `semantics`
- `usage`

24.3.

Figura 24.2.


Profile Configuration
Java
UML Profile for Java
java
lang
Boolean
Byte
Char
Double
Float
Integer
Long
Object
Short
String
math
BigDecimal
BigInteger
net
URL
util
Collection
Iterator
List
Set
SortedSet
Date
Time
Vector
boolean
byte
char
double
float
int
long
short
void

24.3.1.

- boolean
- byte
- char
- double
- float
- int
- long
- short
- void

Nota

24.3.2.

24.3.2.1.

- Boolean
- Byte
- Char
- Double
- Float
- Integer
- Long
- Object
- Short
- String

24.3.2.2.

- Big Decimal
- Big Integer

24.3.2.3.

- URL

24.3.2.4.

- Date
- Time
- Vector

24.3.3.

- Collection
- Iterator
- List
- Set
- SortedSet

A

C

E

G

H

I

J

M

O

P

R

S

T

U

V

W

X

Apéndice A.

A.1.

A.2.

A.2.1.

A.2.2.

A.2.2.1.

A.2.3.

Apéndice B.

B.1.

B.2.

B.2.1.

B.3.

Apéndice C.

C.1.

C.2.

Apéndice D.

D.1.

:
:

D.1.1.

```
header_incl
```

Nota

```
source_incl
typedef_public
typedef_protected
typedef_private
typedef_global_header
typedef_global_source
TemplatePath
email
author
```

Nota

D.1.2.

```
pointer
reference
usage
MultiplicityType
set
get
```

D.1.3.

D.1.3.1.

Aviso

D.1.3.2.

pointer

reference

D.1.4.

D.1.4.1.

cpp_virtual_inheritance

cpp_inheritance_visibility

D.1.5.

D.1.5.1.

cpp_inheritance_visibility

D.1.6.

```
 function Testclass::Testclass()
 // section -64--88-0-40-76f2e8:ec37965ae0:-7fff begin
{
}
 // section -64--88-0-40-76f2e8:ec37965ae0:-7fff end



 void newOperation(std::string test = "fddsaffa")
 // section 603522:ec4c7ff768:-7ffc begin
{
}
 // section 603522:ec4c7ff768:-7ffc end
```

Apéndice E.

E.1.

E.2.

Apéndice F.

This Appendix contains the applied version of the license of this User Manual i.e. the Open Publication License v1.0 with. The latest version is presently available at http://www.opencontent.org/openpub/ [http://www.opencontent.org/openpub/].

F.1.

F.2. Copyright

F.3.

F.4.

1.
2.
3.
4.
5.

F.5.

1.
2.
3.

Apéndice G.

G.1.

G.2.

G.3.

G.4.

Índice

P

R

S

T

V

X