Proyecto de Taller V

Estudio de herramientas para el desarrollo Orientado a Objetos

Informe Principal
Febrero de 2000

Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Autor: Andrés Vignaga
Tutores: Ing. Rodolfo Paiz
Dr. Ing. Alvaro Tasistro

Resumen

En el tema de ingeniería de software orientada a objetos, existen varias metodologías propuestas por distintos autores. El único nivel de estandarización en el área es referente a la notación y es el *Unified Modeling Language*. Este es un lenguaje de especificación que no se ata a ninguna metodología en particular. Debido a su relativamente reciente aparición, no hay una gran variedad de metodologías que lo exploten en profundidad.
Existe especial interés en explorar las potencialidades del *UML* conjuntamente con una revisión de las metodologías orientadas a objetos existentes que permita una propuesta integradora.
En este trabajo se realiza una exploración de las potencialidades del *UML*, aplicado a una metodología de desarrollo orientada a objetos desarrollada en base a la investigación de las existentes.
El resultado es una propuesta metodológica que toma lo principal de los procesos estudiados y que hace uso extensivo de los conceptos manejados en el *UML*; una serie de pautas y consejos para la aplicación del lenguaje en esta metodología, y el desarrollo completo de un caso de estudio basado en todo lo anterior.

Palabras clave: UML, proceso de desarrollo, orientación a objetos, casos de uso, ingeniería de software.

Contenido

Parte I

Introducción

Capítulo 1
Presentación

1.1 Introducción

En este documento se presenta el proyecto de Taller V "Estudio de herramientas para el desarrollo orientado a objetos". En él se plantean los objetivos perseguidos, se explica el abordaje del proyecto, se sintetizan los elementos producidos y se exponen las conclusiones.

1.2 Objetivos del Proyecto

Los objetivos de este proyecto son:

- Explorar las potencialidades del *UML* aplicado al desarrollo de sistemas de software orientado a objetos.
- Estudiar metodologías de desarrollo orientadas a objetos.
- Resumir una metodología que explote las potencialidades del *UML*.
- Aplicar esta metodología a un caso de estudio (sistema de información para el In.Co.).

1.3 Alcance del Proyecto

Para el desarrollo de este proyecto se determinaron las siguientes restricciones:

- El *UML* será estudiado en toda su amplitud, pero la investigación de su uso estará centrado en la especificación del desarrollo de sistemas de software orientados a objetos.
- Las metodologías estudiadas serán únicamente aquellas con un enfoque orientado a objetos.
- Tanto para las metodologías estudiadas como para la metodología descrita, así como su aplicación al caso de estudio, las actividades tratadas son referentes al análisis, diseño e implementación. También están contempladas dentro del alcance, pero en forma superficial, aquellas actividades previas necesarias para las mencionadas anteriormente.

1.4 Logros

- Descripción de una metodología orientada a objetos que hace uso intensivo del *UML*.
- Conocimiento detallado y profundo de *UML*.
- Conocimiento de sus usos respecto de la metodología presentada.
- Construcción de un caso de estudio basándose en la metodología presentada y en los usos del *UML* estudiados.
- Ratificación de la metodología presentada y los usos del *UML*, basada en la experiencia adquirida en desarrollo del caso de estudio.
- Documentación completa de los resultados.

1.5 Desarrollo del Proyecto

El desarrollo del proyecto implicó llevar a cabo una serie de actividades que se describen a continuación. El primer paso consistió en la determinación de las metodologías de interés y su posterior estudio. La siguiente actividad fue el estudio del *UML* desde un punto de vista sintáctico, para luego volver sobre las metodologías y así establecer la conexión entre las actividades de un proceso con el lenguaje. El siguiente paso fue la concepción concreta de la metodología descrita con la aplicación del *UML*. Por último se encaró el desarrollo del caso de estudio en paralelo con el ajuste y documentación de la metodología en forma incremental.

1.6 Documentación del Proyecto

1.6.1 Organización de documentos

La documentación del proyecto está compuesta por cuatro ítems: un documento metodológico que contiene la descripción a nivel macro del proceso de desarrollo, un documento que contiene los detalles del uso del *UML* en el contexto de ese proceso de desarrollo, un documento técnico que contiene lo referente a la aplicación de lo expresado en los dos anteriores al caso de estudio, y un informe principal, que es este documento. La figura 1-1 muestra un diagrama de componentes que permite visualizar esta estructura.

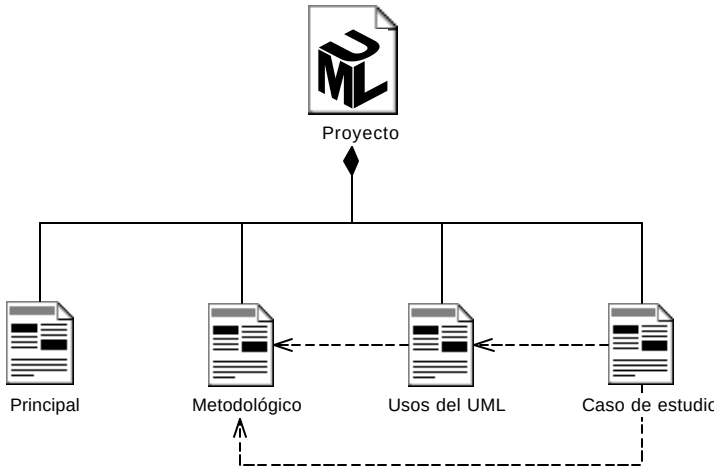


Fig. 1-1: Documentos del proyecto

1.6.2 Recomendaciones para la lectura

Las siguientes recomendaciones son útiles para una fácil lectura de los documentos:

1. Estar familiarizado con la notación de UML.

2. Lectura completa de este documento. Con ello se obtiene:

   - Una sinopsis de UML y una idea de los elementos a tratar en lo sucesivo.
   - Una visión primaria de la metodología.
   - Conceptos fundamentales usados en el proyecto.

3. Lectura del documento metodológico. Con ello se obtiene:

   - Una visión detallada de la metodología.

4. Lectura del documento del caso de estudio teniendo como referencia al documento de usos del *UML*. Con ello se obtiene:

   - Ejemplo de una aplicación de la metodología a un caso concreto.
   - Pautas de uso de los elementos del *UML*.
   - Detalles de construcción de los diagramas del *UML*.
   - Ejemplificación del uso de *UML* según la metodología descrita.

1.6.3 Estructura de los documentos

Todos los documentos generados están estructurados de una misma manera. Tienen una primera parte que corresponde a una introducción, en la que se hace una presentación del documento y se explica cómo está organizado. El contenido central puede estar formado por una o dos partes. Por último, una parte final contiene los apéndices.

Para facilitar la lectura, los temas están en general organizados en capítulos cortos, con la única excepción de los capítulos correspondientes a los ciclos de desarrollo del documento del caso de estudio. Por su parte, existe una correspondencia explícita entre cada tema tratado en el documento de metodología con su aplicación en el documento de caso de estudio. Esta correspondencia está mostrada en la tabla 2-1 del documento metodológico.

Los apéndices de los restantes documentos contienen un glosario específico al documento. En éste no se incluye uno por considerarse redundante.

Capítulo 2
Organización del Documento

Este documento es el informe principal del proyecto de Taller V "Estudio de herramientas para el modelado orientado a objetos" y contiene información general del proyecto, como ser una presentación, los objetivos y las conclusiones, así como capítulos dedicados a los temas específicos tratados durante su desarrollo.

Este documento está organizado de la siguiente forma: está dividido en tres partes, la primera corresponde a una introducción, la segunda a las actividades desarrolladas en el proyecto y la tercera a los apéndices.

La segunda parte contiene información acerca de las actividades desarrolladas en el proyecto: el capítulo 3 está dedicado a presentar el *Unified Modeling Language*. El capítulo 4 contiene un resumen de la metodología descrita en el documento metodológico. En el capítulo 5 se presenta el caso de estudio y se tratan temas de su desarrollo. El capítulo 6 está dedicado a las conclusiones a las que se llegó luego de completado el proyecto y al planteo de los trabajos futuros.

La tercera parte contiene un apéndice en el que se presenta con cierto detalle dos de los conceptos más importantes que se usaron en el proyecto, casos de uso y contratos de software. La inclusión de estos temas en este documento es debido a su importancia y a que no tienen por qué ser familiares para el lector.

Parte II

El Proyecto

Capítulo 3
Unified Modeling Language

3.1 Introducción

En este capítulo se presenta al *Unified Modeling Language* y se describen sus principales elementos. El *UML* es un lenguaje para especificar, construir, visualizar y documentar la información producida en un proceso de desarrollo.
Este lenguaje fue desarrollado por Grady Booch, James Rumbaugh e Ivar Jacobson. Es el resultado de la *unificación* de la notación y de los conceptos de las metodologías de sus tres autores, *Booch Method, OMT* y *OOSE* respectivamente.

3.2 Evolución del *UML*

A fines de la década de los 80's y principios de los 90's, existían numerosas propuestas metodológicas en el área del desarrollo orientado a objetos, cada cual con su propia notación. Las más destacadas eran el *Booch Method* de Grady Booch, *OMT* de James Rumbaugh y *OOSE* de Ivar Jacobson. Cada una de ellas era una metodología completa y eran reconocidas por tener cada una puntos fuertes en determinadas áreas. *OOSE* se mostraba más propicia para desarrollar ingeniería de requerimientos, *OMT* era potente para el análisis y el *Booch Method* para el diseño.
A mediados de los 90's Booch y Rumbaugh trabajando para la misma empresa decidieron unificar sus metodologías en lo que se llamó *Unified Method*. Algún tiempo después, Jacobson se unió al equipo y aportó su metodología al proyecto. Esta unificación de metodologías resultó complicada por lo que decidieron tomar los conceptos fundamentales de cada una y las notaciones, para crear un *lenguaje* de modelado unificado. Así surge la primera versión del *UML*, la que es hecha pública y es bien recibida por la comunidad de desarrolladores.
Ante esto, el *Object Management Group* (OMG) en un *request for proposal* captan la atención de las principales empresas del medio, las que crean un grupo conjunto de trabajo. El resultado final es la versión 1.1 del *UML* el cual es adoptado por la *OMG* en 1997 como lenguaje estándar de modelado. Actualmente la última versión es la 1.3 y se está trabajando en la 2.0.

3.3 Características del *UML*

El UML es un lenguaje de modelado cuya finalidad es modelar sistemas de software usando conceptos del paradigma de orientación a objetos. La idea principal que manejaron sus autores fue la de crear una herramienta lo suficientemente rica en semántica y notación que permita expresar la gran mayoría de los conceptos fundamentales de amplia aceptación y difusión en la comunidad de desarrolladores de sistemas de software orientados a objetos.

La principal característica de este lenguaje es que es independiente de todo lenguaje de programación y **fundamentalmente** de todo proceso de desarrollo. Esto último tiene estrecha relación con la idea mencionada en el párrafo anterior y explica la aceptación masiva en la comunidad de desarrolladores (aún antes de que la *OMG* aceptara al *UML* como un estándar). La experiencia mostró que no es posible estandarizar *un* proceso de desarrollo multipropósito, por lo cual en su lugar se estandarizó *una* notación.

3.4 El *UML*: su notación y su uso

Como se dijo antes, el *UML* es una herramienta de modelado y no guía al desarrollador en cómo realizar un análisis y diseño orientado a objetos, o cuál proceso de desarrollo utilizar.
Considerando que el *UML* es sólo un lenguaje, el hecho de conocer su semántica y notación no implica saber darle uso. Usar el *UML* consiste en crear modelos que representen cierta información de utilidad generada durante el desarrollo de un producto de software. Para saber qué información generar para luego expresarla usando el lenguaje se requiere de un ingrediente fundamental: *una metodología.*
Adoptando una metodología concreta, por un lado el desarrollador conoce exactamente qué piezas de información debe generar en cada caso, y por otro, es posible además desarrollar *técnicas* de modelado. Ayudado por estas técnicas, el desarrollador es capaz de expresar en forma eficiente en *UML* la información que exige la metodología. Pero eso no es todo, un enfoque podría ser generar completamente esa información mentalmente y luego modelarla, pero dada la característica visual del lenguaje, si el generar la información se hace conjuntamente con el modelado, el proceso de generar *esa* información se ve facilitado.
De esa forma, el *UML* no solamente es un lenguaje para visualizar, especificar y documentar, sino que es también un lenguaje de construcción.

En resumen, hay dos aspectos importantes a destacar. Primero, no tiene sentido hablar de aplicar el *UML* en solitario y sin una metodología. Segundo, dada una metodología es posible pensar técnicas de modelado que por un lado hagan que el *UML* ayude a aplicar la metodología y por otro faciliten la documentación de los resultados.

En el capítulo 4 se describe superficialmente una metodología que es explicada en forma detallada en el documento metodológico. En el documento de usos del *UML* se tratan técnicas de modelado respecto de la metodología presentada y por último, se muestra la aplicación de todo esto en el documento de caso de estudio.

3.5 Estructura del *UML*

El *UML* está compuesto por ocho tipos de diagramas y un conjunto de elementos generales. Estos elementos generales son los adornos, los mecanismos de extensión, los packages y el *OCL*. Esta estructura se puede visualizar en el diagrama de componentes mostrado en la figura 3-1.

3.5.1 Diagramas

El *UML* ofrece ocho tipos distintos de diagramas, ellos son:

- Diagrama de estructura estática, que muestra el esqueleto estable de una aplicación en términos de clases y sus relaciones.
- Diagrama de componentes, que muestra posibles configuraciones de distintos tipos de componentes de software.

- Diagrama de deployment, que muestra la topología de un sistema y los elementos de hardware que funcionan en tiempo de ejecución.
- Diagrama de casos de uso, que muestran los casos de uso de una aplicación así como los actores y sus relaciones.
- Diagrama de secuencia, que muestra un conjunto de mensajes enviados entre objetos (una interacción), enfatizando su organización según su ocurrencia en el tiempo.
- Diagrama de colaboración, que muestra también una interacción, pero enfatizando las relaciones entre los objetos que en ella participan.
- Diagrama de estados, que muestra los cambios de estados que sufre un elemento a lo largo de su ciclo de vida.
- Diagrama de actividad, que muestra los pasos que son necesarios llevar a cabo para la realización de una cierta actividad o proceso.

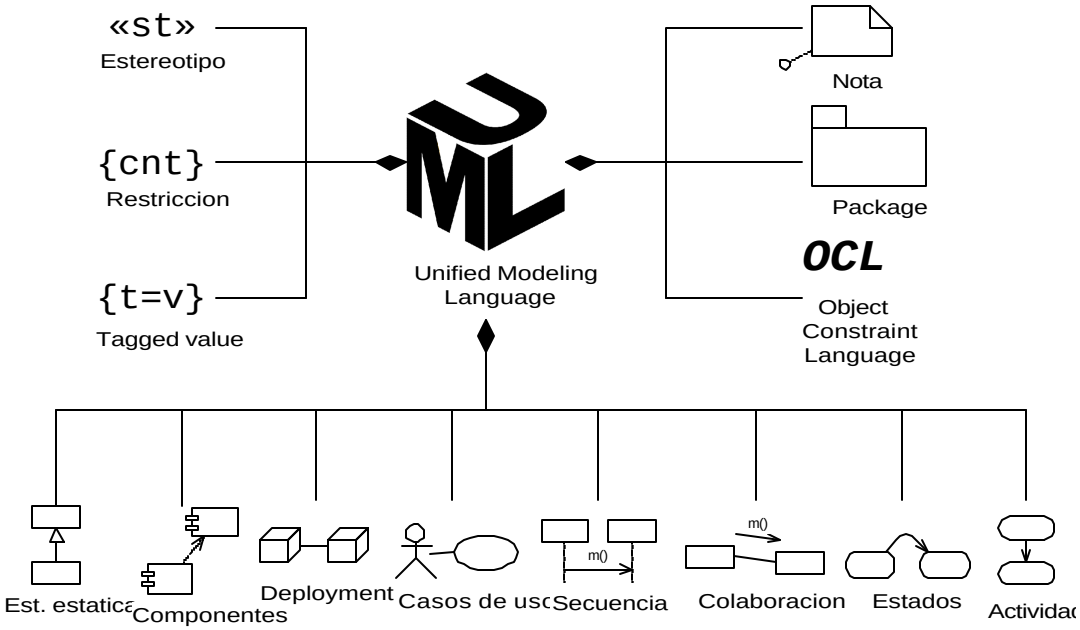


Fig. 3-1: Estructura del *UML*

3.5.2 Adornos

Los adornos son gráficos o texto que se agregan a la representación gráfica básica de un elemento para visualizar detalles de su especificación.
Existen varios tipos de adornos, pero las notas son el adorno más común. Una nota está asociada a un elemento o a un grupo de elementos y expresa un comentario o una restricción.

3.5.3 Mecanismos de extensión

Los mecanismos de extensión del *UML* permiten extender el lenguaje en una forma controlada. Existen tres mecanismos de extensión: estereotipos, tagged values y restricciones.

3.5.3.1 Estereotipos

Un estereotipo es una extensión del vocabulario del *UML* y extiende la semántica de su metamodelo, permitiendo construir nuevos elementos, a partir de elementos ya existentes sin modificar su estructura, pero específicos para el problema en desarrollo.

3.5.3.2 Tagged values

Mediante los tagged values es posible agregar propiedades a los elementos del *UML*. Cada tag referencia a una propiedad determinada de un elemento, a la que se le asigna un valor.

3.5.3.3 Restricciones

Las restricciones son condiciones semánticas aplicadas a elementos, que deben satisfacerse en todo momento en un modelo para que el elemento esté bien formado.

3.5.4 Packages

El *UML* incluye los packages que son un mecanismo de propósito general para agrupar elementos. Los packages pueden ser usados para organizar elementos con cualquier propósito. El criterio a usar para determinar esta organización no está definido en el *UML*.

3.5.5 OCL

El *Object Constraint Language* es un lenguaje formal de expresiones sin efectos laterales sobre su entorno, que fue desarrollado con el objetivo de dar una herramienta de especificación de restricciones no ambiguas sobre objetos.

Capítulo 4
Metodología

4.1 Introducción

En este capítulo se describe una metodología que define un proceso de desarrollo según un enfoque orientado a objetos, basado en casos de uso y correspondiente a un modelo iterativo e incremental.

4.2 Características y Macroetapas

Esta metodología está basada en la propuesta *RPM* del autor Craig Larman, pero también complementada con el *Booch Method*, *OMT* y *OOSE*, y con algunos aportes de la propia experiencia del autor de este proyecto. En esencia se centra en actividades de análisis, diseño e implementación orientada a objetos, siguiendo un modelo iterativo e incremental de desarrollo. Según este modelo, la construcción del sistema se divide en partes y cada una de ellas es atacada y desarrollada por completo de a una por vez. Esto requiere de dos elementos. Por un lado, previo a la construcción, es necesario realizar una planificación y elaboración del problema, en la que entre otras cosas se determina la división que condicionará la etapa de construcción. Por otro lado, es necesario contar con un criterio bien definido mediante el cual se determine esta división.
De esta manera se definen dos macroetapas: una primera de planificación y elaboración, y otra de construcción.

El criterio de partición del problema a usar son los casos de uso. En la etapa de planificación y elaboración se identifican todos los casos de uso del sistema y en la etapa de construcción se implementa el comportamiento de los casos de uso, generalmente uno en cada ciclo de desarrollo. Resulta evidente, pues, el impacto que tiene sobre la etapa de construcción la identificación de los casos de uso y sobre todo el proceso de determinar en qué ciclo atacar a cada uno.

4.3 Planificación y Elaboración

Esta es una etapa puramente de investigación y análisis, donde el objetivo es principalmente obtener un conocimiento de la realidad del problema, definir qué es lo que se debe hacer y planificar la etapa siguiente (construcción) basándose en el enfoque mencionado anteriormente.
En la figura 4-1 se muestra un diagrama de actividad que modela las actividades a realizar en esta etapa. Cabe aclarar que el ordenamiento de estas actividades puede variar respecto del presentado en el diagrama.

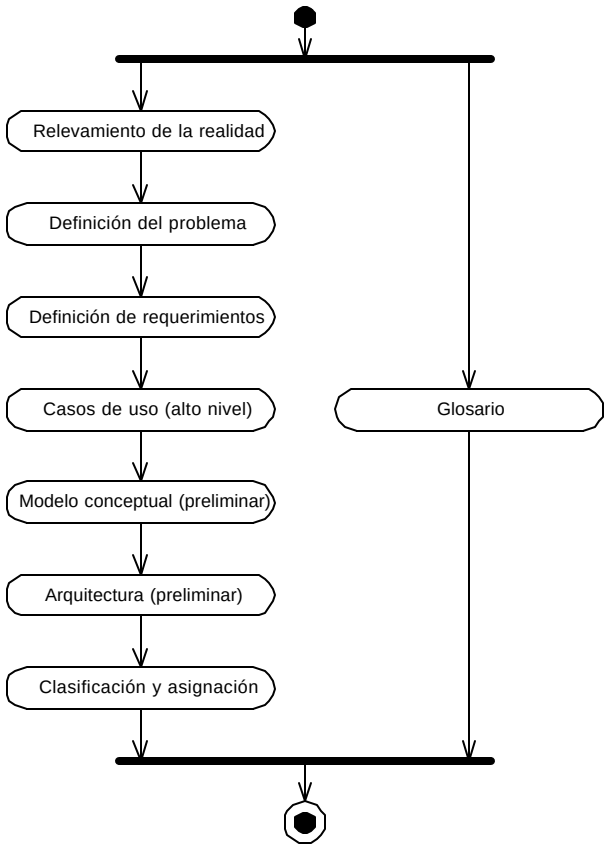


Fig. 4-1: Actividades de planificación y elaboración

A continuación se describen las actividades dentro de la etapa de planificación y elaboración.

4.3.1 Relevamiento de la realidad

Esta actividad consiste en investigar el dominio del problema, obteniendo una descripción lo más completa posible de la realidad. También se estudian los principales procesos que se llevan a cabo.

4.3.2 Definición del problema

Esta actividad consiste en determinar en forma clara y precisa el problema a resolver. También se expresa la información general del producto a desarrollar, del tipo atributos, restricciones, dependencias, etc.

4.3.3 Definición de requerimientos

Esta actividad consiste en especificar en una forma no ambigua los requerimientos funcionales del producto a desarrollar, es decir, declaraciones de lo que se espera que el sistema haga.

4.3.4 Casos de uso de alto nivel

Esta actividad consiste en identificar los actores y casos de uso del sistema, y realizar una especificación de los mismos en alto nivel. Durante esta actividad es que se produce el particionamiento del sistema, que será utilizado más adelante para planificar la etapa de construcción.
Dada la importancia del concepto de caso de uso, este se explica con más detalle en el apéndice A.

4.3.5 Modelo conceptual preliminar

Esta actividad consiste en identificar los conceptos relevantes del dominio de la aplicación, junto con sus relaciones. Esta abstracción se realiza a partir de la descripción resultante del relevamiento de la realidad y busca profundizar el conocimiento que se tiene del problema.

4.3.6 Arquitectura preliminar

Esta actividad consiste en, por un lado, identificar subsistemas de forma de dividir el sistema en unidades de menor complejidad y tamaño, y por otro, expresar una versión de alto nivel de la arquitectura lógica del sistema con un fin similar al anterior, de crear divisiones (eventualmente ortogonales a las anteriores) para reducir la complejidad.

4.3.7 Clasificación y asignación de casos de uso

Esta actividad consiste en determinar cuál caso de uso será atacado en determinado ciclo de desarrollo. La clasificación consiste en ordenar los distintos casos de uso según algún criterio (por ejemplo, complejidad) y la asignación, simplemente hacer corresponder los casos de uso en forma correlativa con los ciclos, según el orden determinado.

4.4 Construcción

La etapa de construcción es donde se produce la iteración en el proceso de desarrollo. Consiste básicamente en iterar un ciclo de desarrollo completo para cada caso de uso identificado en la etapa de planificación y elaboración. Cual caso de uso se ataca en la iteración *i* fue determinado previamente en la clasificación y asignación. En la figura 4-2 se muestra un diagrama de actividad que modela las actividades a desarrollar en cada uno de los ciclos iterativos. Los andariveles organizan las actividades según sean de análisis, diseño o implementación.

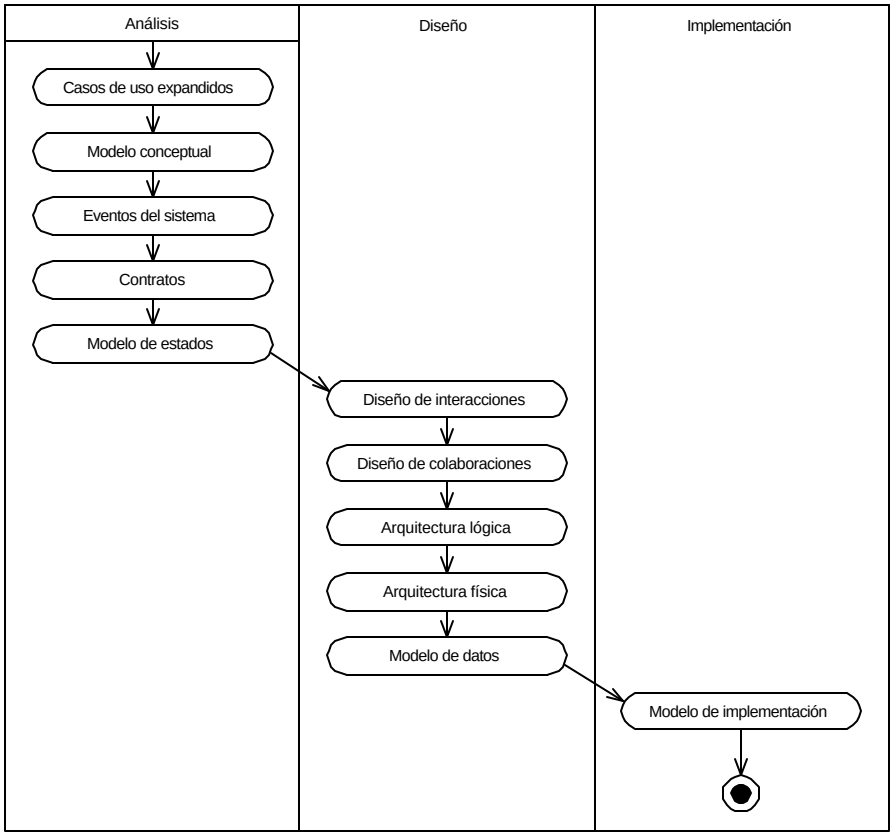


Fig. 4-2: Actividades de un ciclo de desarrollo

A continuación se describen las actividades dentro de un ciclo de iteración.

4.4.1 Casos de uso expandidos

Esta actividad consiste en realizar una especificación más detallada del caso de uso a atacar en el ciclo (recordar que estaba especificado en alto nivel). Una especificación como esa se denomina expandida (ver apéndice A). Su cometido es mostrar todo el detalle necesario para el desarrollo del caso de uso.

4.4.2 Modelo conceptual

Esta actividad es similar a la realizada (para toda la realidad) en la etapa anterior, pero en este caso en un nivel mayor de detalle, y solamente aplicado a aquella parte de la realidad que tenga que ver con el caso de uso a tratar.

4.4.3 Eventos del sistema

Esta actividad consiste en identificar aquellos eventos generados por los actores sobre el sistema durante el transcurso del caso de uso tratado. El propósito de esto es reflejar directamente el diálogo entre el sistema y los actores en el caso de uso.

4.4.4 Contratos de software para operaciones del sistema

Esta actividad consiste en especificar mediante un contrato de software las operaciones del sistema. Las operaciones del sistema son operaciones *internas* al sistema que se disparan automáticamente con la ocurrencia de eventos del sistema. El resultado de estas operaciones es el resultado que el actor espera.
Dada la importancia del concepto de contrato de software, este se explica con más detalle en el apéndice A.

4.4.5 Modelo de estados

Esta actividad consiste en expresar los estados por los que pasa un determinado elemento durante su ciclo de vida y qué eventos disparan las transiciones entre esos estados.

4.4.6 Diseño de interacciones

Esta actividad consiste en determinar qué entidades de software colaboran (y de qué manera) para resolver las operaciones del sistema. Esta actividad está estrechamente relacionada con la asignación de responsabilidades[1] y es una de las más delicadas del desarrollo. En definitiva, se busca determinar un conjunto de objetos que intercambiando mensajes (interacción) obtengan como resultado el efecto especificado en las postcondiciones de los contratos de las operaciones del sistema.

4.4.7 Diseño de colaboraciones

Esta actividad consiste en modelar las clases de objetos (junto con sus relaciones) que participan en las interacciones que resuelven las operaciones del sistema. Esto recibe el nombre de colaboración y se dice que realiza el caso de uso tratado. El propósito de esto es tener un modelo de las clases que deberán ser implementadas de forma de poder pasar directamente a la codificación.

4.4.8 Arquitectura lógica

Esta actividad consiste en refinar la arquitectura propuesta en la etapa de planificación y elaboración con aquellos elementos de diseño que hayan aparecido por primera vez en el presente ciclo de desarrollo. El propósito de esto es, nuevamente, agrupar elementos en forma cohesiva y con bajo acoplamiento entre grupos.

4.4.9 Arquitectura física

Esta actividad consiste en determinar el entorno físico de ejecución de la aplicación. El propósito es mostrar la topología de la aplicación, haciendo énfasis en los elementos presentes en tiempo de ejecución.

4.4.10 Modelo de datos

Esta actividad consiste en identificar aquellos objetos que necesiten ser persistentes y determinar la estructura lógica y física de la base de datos en la que se almacenarán estos objetos. También se determina el medio a través del cual los objetos pasan de memoria a la base y viceversa.

---

[1] Un conjunto adecuado de criterios de asignación de responsabilidades es el presentado con el nombre de *GRASP* por Larman-97 en "Applying UML and Patterns". Prentice-Hall.

4.4.11 Modelo de implementación

Esta actividad consiste en documentar los aspectos de la implementación referentes al código fuente implementado, así como las configuraciones de ejecutables y otros elementos necesarios en tiempo de ejecución que estén en relación con la especificación de la arquitectura física previamente realizada.

4.5 Actividades y Modelos

A continuación se muestra la relación entre las actividades del proceso descrito y los elementos del *UML* usados para su especificación. Para ver en forma más profunda la aplicación de estos elementos a las actividades del proceso, ver el documento de usos del *UML*.

4.5.1 Planificación y elaboración

La figura 4-3 muestra la relación entre las actividades de esta etapa con los elementos del *UML* que se usan para su modelado.

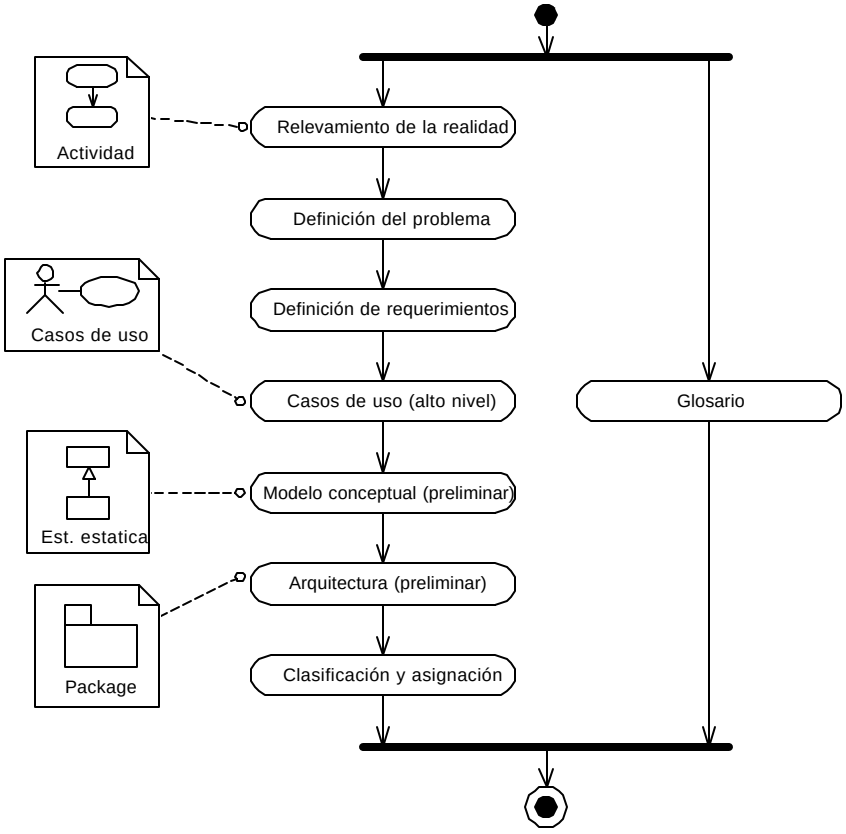


Fig. 4-3: Elementos usados para el modelado en la planificación y elaboración

4.5.2 Construcción

4.5.2.1 Análisis

La figura 4-4 muestra la relación entre las actividades de esta etapa de análisis con los elementos del *UML* que se usan para su modelado.

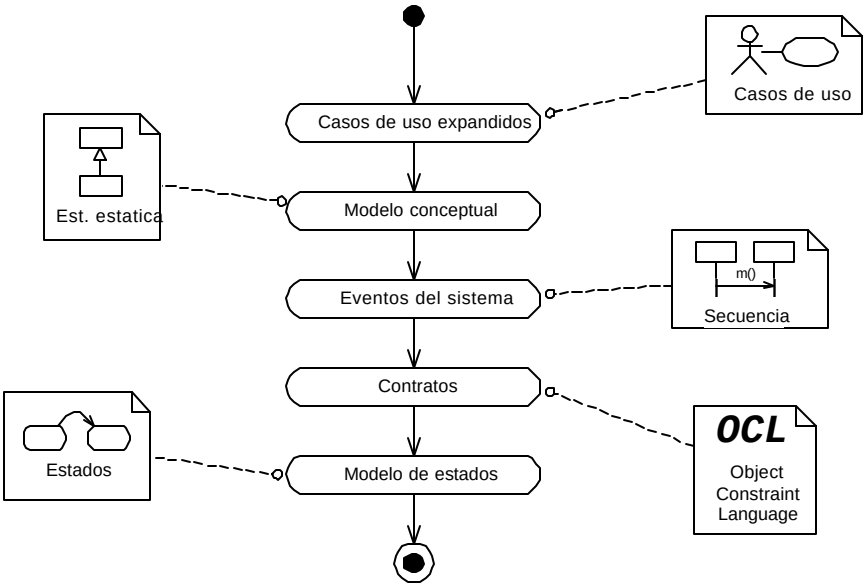


Fig. 4-4: Elementos usados para el modelado en el análisis

4.5.2.2 Diseño

La figura 4-5 muestra la relación entre las actividades de esta etapa de diseño con los elementos del *UML* que se usan para su modelado.

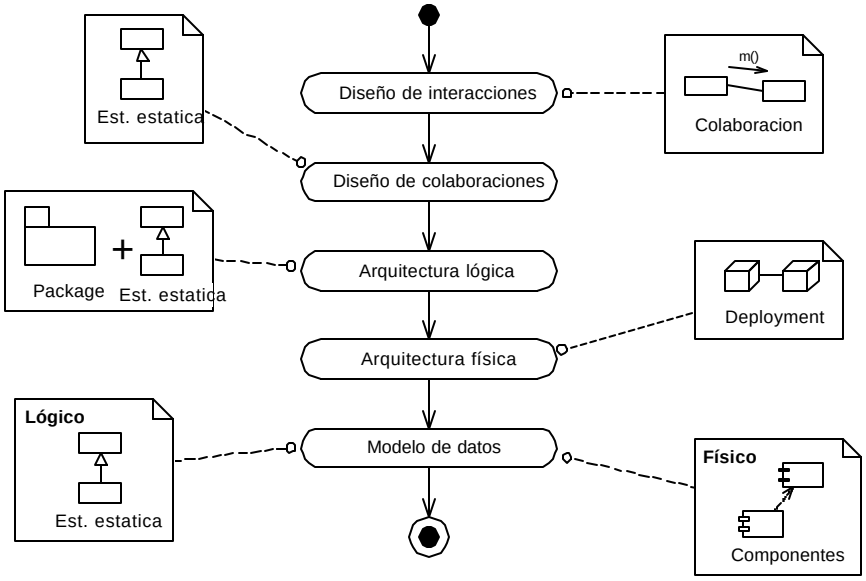


Fig. 4-5: Elementos usados para el modelado en el diseño

4.5.2.3 Implementación

La figura 4-6 muestra la relación entre las actividades de esta etapa de implementación con los elementos del *UML* que se usan para su modelado.

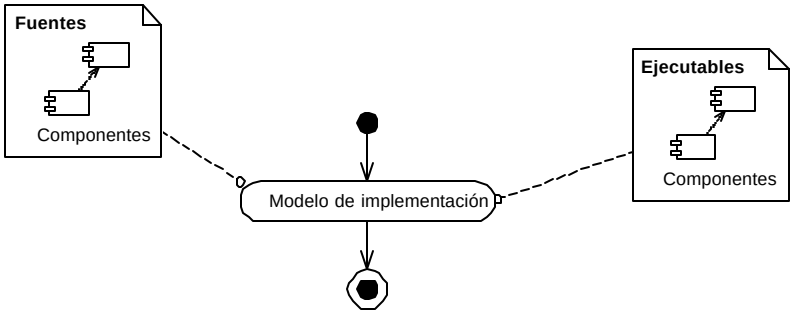


Fig. 4-6: Elementos usados para el modelado en la implementación

Capítulo 5
Caso de Estudio

5.1 Introducción

En este capítulo se presenta el caso de estudio desarrollado y se mencionan aspectos de su desarrollo. En primera instancia se describe sumariamente el problema, (por detalles, ver el documento de caso de estudio). A continuación se ilustran los detalles más relevantes de aspectos de planificación, análisis y diseño. Finalmente se mencionan aspectos de la implementación.

5.2 El Problema

El Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República (In.Co.) dispone de un presupuesto el cual es dedicado al mantenimiento del plantel docente que en él desarrolla actividades. El mantenimiento del plantel docente incluye básicamente: contratación de cargos y cambio de la carga horaria de dedicación.

El sistema de software a desarrollar es "Sistema de Presupuesto Docente del In.Co.". Este sistema deberá automatizar la gestión del presupuesto docente del Instituto de Computación de la Facultad de Ingeniería.

El presupuesto en el Instituto es fijo y en un principio igual todos los meses. Esto significa que para crear un cargo u otorgar un aumento en la carga horaria de un docente es necesario contar con disponibilidad de recursos.
Las disponibilidades son fondos no utilizados para algún cargo o se generan al producirse la renuncia o disminución en la carga horaria de un docente
Debido a que el presupuesto es cerrado, es fundamental llevar un control detallado de las disponibilidades. A su vez cuando se quiera realizar un gasto (contratación o ampliación) es necesario indicar con qué recursos se financiará el mismo, y se los debe detallar indicando antecedentes de las disponibilidades, por ejemplo si es por un cargo vacante, el nombre del docente que lo ocupaba. Es necesario que el valor del conjunto de los recursos, se ajuste lo más exactamente posible al costo del gasto a financiar.
Por lo expresado, el proceso de determinar la forma de financiación de un determinado gasto resulta complicado.

El sistema comprende además la gestión de cargos transitorios y movimientos transitorios de cargos presupuestales, que por su parte son financiados con otro tipo de recursos (economías). Para este tipo de recurso, se aplican las mismas consideraciones que para las disponibilidades presupuestales.

En resumen, el objetivo es automatizar la gestión del presupuesto docente del In.Co., tarea que actualmente se realiza en forma manual. Específicamente se pretende:

- Automatizar los procesos de contratación, renuncia, aumento y disminución de la carga horaria de un cargo docente.
- Brindar una manera simple y ágil de generar la financiación de un gasto a partir de los datos almacenados, simulando el uso de distintos ítems de disponibilidad.

5.3 El Desarrollo

Para el desarrollo del caso de uso se utilizo la metodología descrita en el capítulo 4 (y explicada en el documento metodológico), aplicando el *UML* como notación y tomando usando como referencia la información contenida en el documento de usos del *UML.*
A continuación se muestran los puntos relevantes de las dos macroetapas del proceso de desarrollo.

5.3.1 Planificación y Elaboración

Aquí se muestra el aspecto de planificación de esta etapa que consiste en un análisis de los requerimientos, las funcionalidades y el comportamiento del sistema que den los elementos para determinar la estructura de la etapa de construcción. Para ello se realizó un análisis de requerimientos y de casos de uso para realizar la clasificación y asignación de casos de uso a ciclos de desarrollo.

5.3.1.1 Requerimientos

Se definieron un total de 42 requerimientos funcionales organizados en tres grandes categorías: presupuestales, transitorios y simulación.

5.3.1.2 Casos de uso

Se identificaron los siguientes actores:

- Direccion
- Administrador
- Consejo
- RRHH (recursos humanos)
- Simulador

y los siguientes casos de uso:

- Crear un cargo docente
- Renuncia a un cargo
- Reducción horaria
- Ampliación horaria
- Crear cargo docente transitorio
- Reducción horaria transitoria
- Ampliación horaria transitoria
- Simulación de pago presupuestal
- Simulación de pago con economías

5.3.1.3 Clasificación y asignación

Con la información contenida en los documentos de casos de uso de alto nivel, se realizó la clasificación de los casos de uso que se muestra en la tabla 5-1.

Ranqueo	Caso de Uso	Justificación
Alto	Crear un cargo docente	Tiene un significante impacto en la arquitectura del sistema. Representa un proceso primario.
	Ampliación horaria	Representa un proceso primario.
	Reducción horaria	Representa un proceso primario.
	Renuncia a un cargo	Representa un proceso primario.
Medio	Simulación de pago presupuestal	Resulta de particular interés para el usuario del sistema.
Bajo	Creación de cargo transitorio	Representa un proceso de menor importancia.
	Ampliación horaria transitoria	Representa un proceso de menor importancia.
	Reducción horaria transitoria	Representa un proceso de menor importancia.
	Simulación de pago con economías	Resulta de particular interés para el usuario del sistema.

Tabla 5-1: Clasificación de casos de uso

La siguiente tabla muestra la asignación de casos de uso a los distintos ciclos de desarrollo:

Ciclo de desarrollo	Caso de uso
1	Creación de cargo docente
2	Ampliación horaria
3	Reducción horaria
4	Renuncia a un cargo
5	Simulación de pago presupuestal
6	Creación de cargo transitorio
7	Ampliación horaria transitoria
8	Reducción horaria transitoria
9	Simulación de pago con economías

Tabla 5-2: Asignación de casos de uso

5.3.2 Construcción

Para la construcción se realizaron fielmente las actividades definidas en la metodología y se produjo una documentación amplia de los resultados. A continuación se usa una versión resumida del caso de uso *Crear un cargo docente* atacado en el ciclo de desarrollo 1 como medio para mostrar los puntos de interés en la etapa de construcción.

5.3.2.1 Análisis

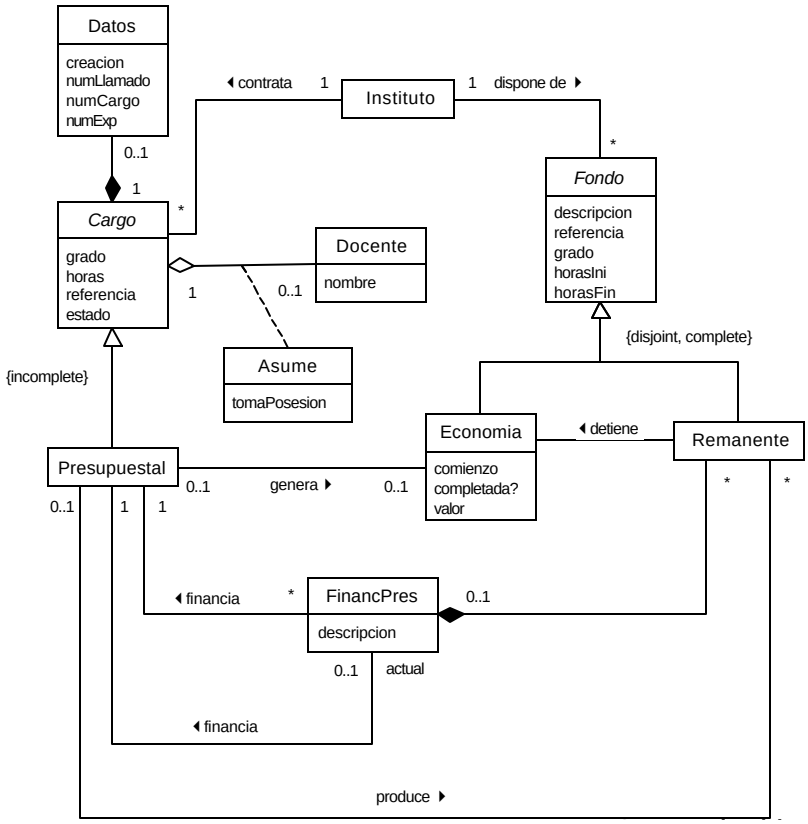
El siguiente es el documento de caso de uso expandido:

Caso de uso:	**Crear un cargo docente**
Actores:	Direccion (iniciador), Administrador, Consejo, RRHH
Propósito:	Reflejar la creación de un cargo docente con financiación presupuestal.
Sinopsis:	La Direccion emite una solicitud para la creación de un cargo docente y el Administrador especifica los remanentes a usar como forma de financiación del mismo. El Consejo aprueba la solicitud y se crea el cargo, se registra los remanentes usados como forma de financiación, el cargo docente comienza a existir. RRHH proporciona la información del docente cuando este toma posesión.
Tipo:	Primario y esencial
Referencias cruzadas:	*Requerimientos*: R1.1.1, R1.1.2, R1.1.3, R1.1.4, R1.1.5, R1.5.2, R1.5.3, R1.5.4, R2.1.1, R2.1.2, R2.1.3, R2.1.4

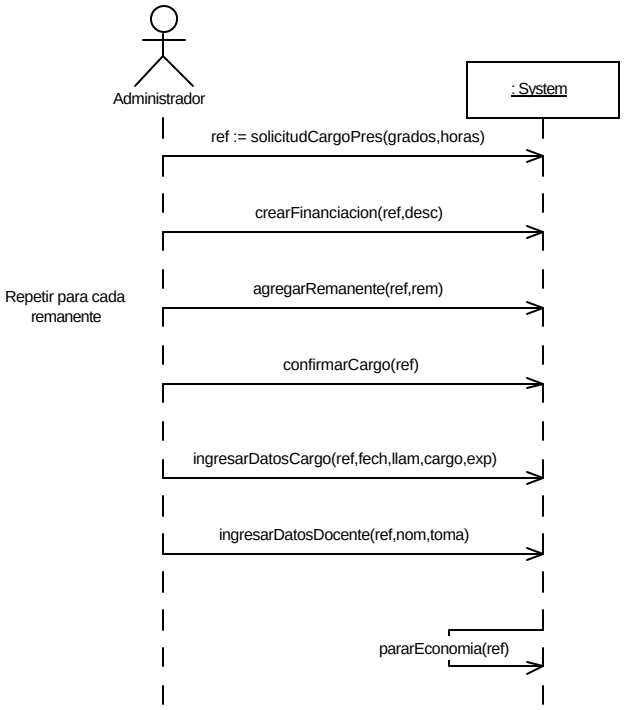
Curso Típico de Eventos

Acción del Actor	Respuesta del Sistema
1. Este caso de uso comienza cuando la Direccion emite una solicitud de creación de cargo docente.	
2. El Administrador ingresa la solicitud dando los detalles del llamado.	
	3. Registra la solicitud.
4. El Administrador proporciona los remanentes que conforman la forma de financiación.	
	5. Registra la financiación.
6. El Consejo aprueba el cargo.	
7. El Administrador confirma el cargo.	
	8. Crea el cargo.
	9. Quema los remanentes especificados en la forma de financiación (los cuales dejan de generar economías).
	10. Genera la economía correspondiente.
11. RRHH proporciona los datos del cargo.	
12. El Administrador ingresa los datos del cargo.	
	13. Registra los datos.
14. RRHH proporciona los datos del docente que asume el cargo.	
15. El Administrador ingresa los datos del docente.	
	16. Deja de generar la economía.
	17. Registra los datos.

El siguiente es el modelo conceptual para este caso de uso:



El diagrama de secuencia del sistema muestra los eventos del sistema para este caso de uso:



De todas las operaciones del sistema, se toma `solicitudCargoPres` y se muestra el contrato de software asociado:

Contrato

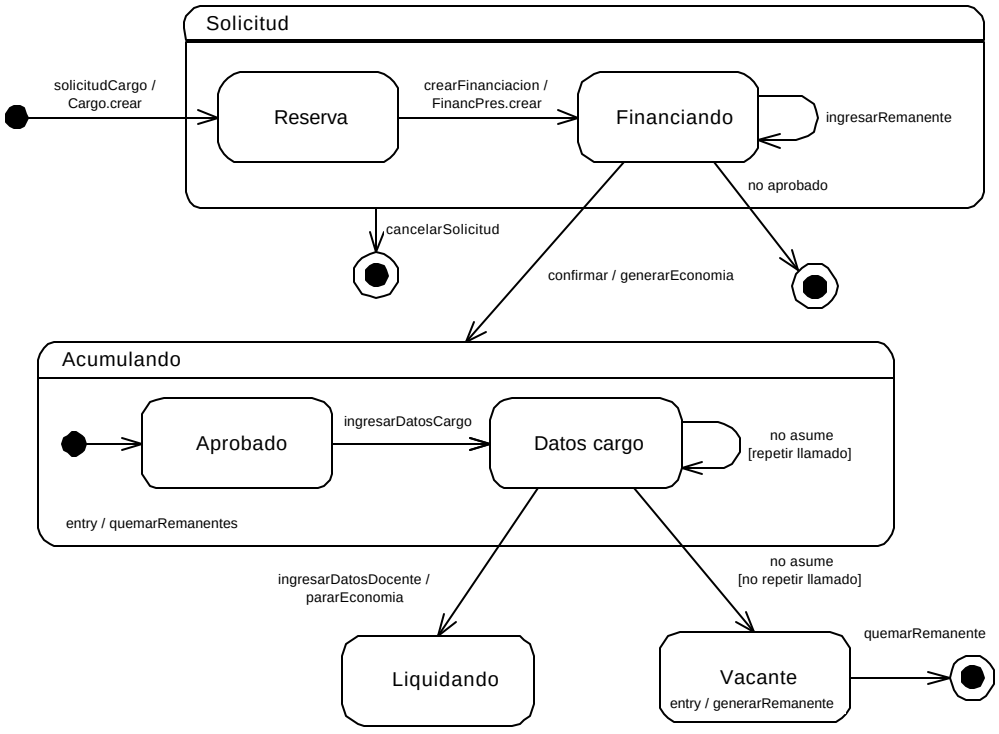
Nombre:	solicitudCargoPres(grado : integer, horas : integer)
Responsabilidades:	Crear una solicitud para un cargo con el grado y la carga horaria especificados.
Tipo:	Sistema
Referencias cruzadas:	*Requerimientos:* R1.1.1 *Casos de uso:* Crear un cargo docente
Notas:	El número de referencia del cargo es generado por el sistema.
Excepciones:	Si el grado o la carga horaria no son válidos, indicar que se produjo un error.
Salida:	Número de referencia del cargo
Precondiciones:	

```
grado >= 1 and grado <= 5 and horas >= 1 and horas <= 41
```

Postcondiciones:

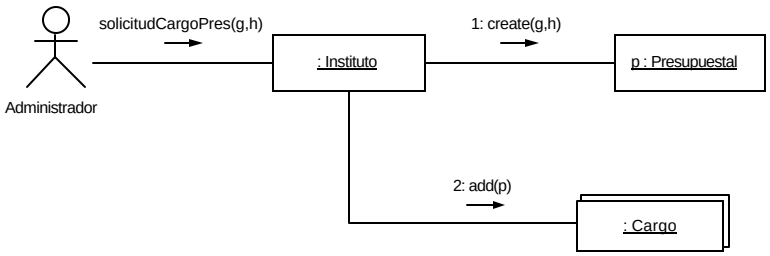
```
Instituto
  self.cargo = self.cargo@pre->including(new p : Presupuestal
      | p.grado = grado and p.horas = horas and
        p.referencia = result and p.estado = #solicitud)
```

El siguiente diagrama de estados muestra el ciclo de vida parcial de un cargo presupuestal en lo referente a su creación (no se consideran ampliaciones, reducciones ni renuncias):

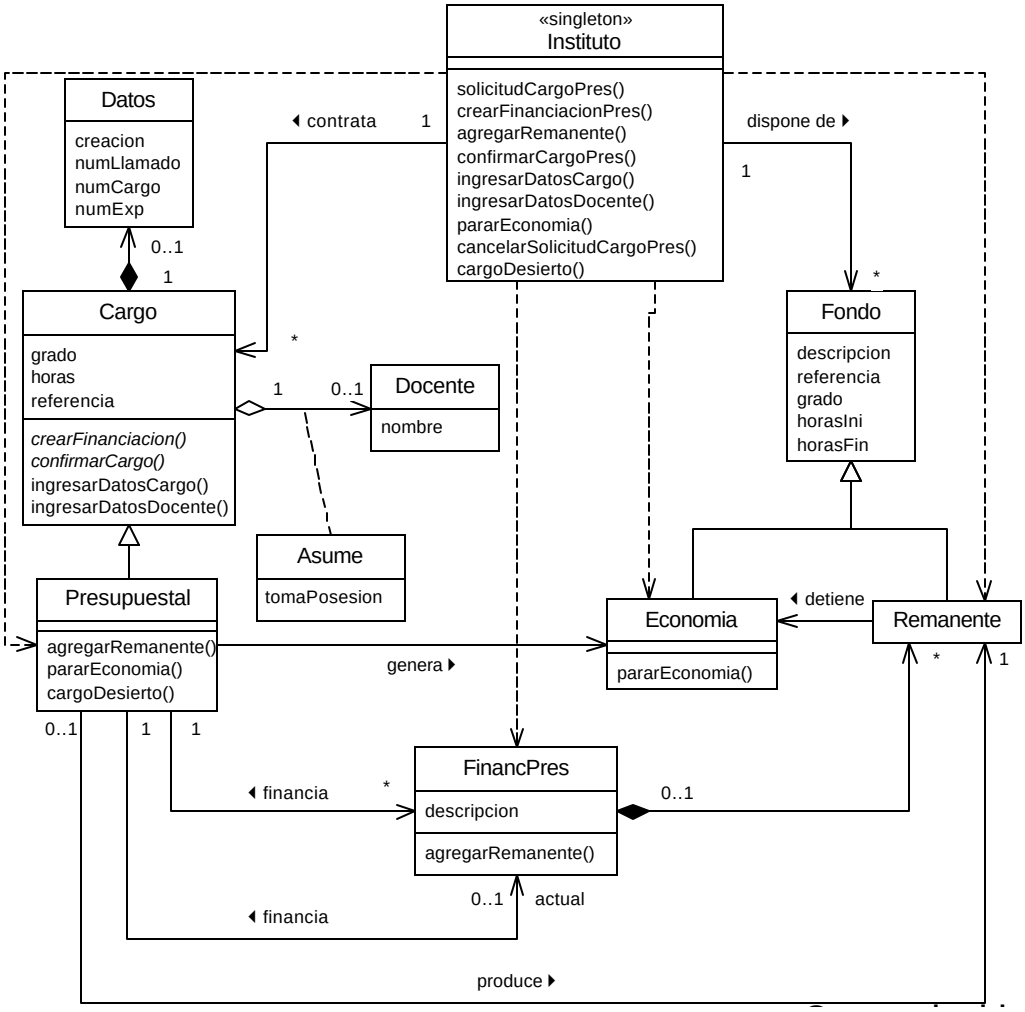


5.3.2.2 Diseño

A continuación se muestra la interacción que da origen cuando es invocada la operación del sistema `solicitudCargoPres`:



El siguiente diagrama de clases muestra la colaboración completa en la que dan lugar todas las interacciones de este caso de uso, en particular la anterior:



5.3.3 Aspectos de la implementación

La codificación de la solución no era un punto relevante en este proyecto, por lo tanto se realizó:

- Por considerarse que la experiencia aportada es de interés.
- Para completar la aplicación de la metodología y contar con elementos reales para documentar en esta etapa.
- Porque existió interés de entregar al Administrador del presupuesto del In.Co. un producto *completo* que le ayude en la gestión del presupuesto.

Esto motivó la elección del lenguaje de programación Microsoft Visual Basic para la codificación. Este lenguaje, si bien no es la herramienta más adecuada para la implementación de aplicaciones desarrolladas con este tipo de metodologías, brinda en forma predefinida elementos que simplifican la tarea de codificación.

Capítulo 6
Conclusiones y Trabajos Futuros

6.1 Conclusiones

Concluido el proyecto, corresponde resaltar:

- Los logros (detallados en la sección 1.4) coinciden ampliamente con los objetivos y resultados esperados mencionados en la propuesta original del proyecto.
- Ver los aspectos metodológicos en su globalidad y los conceptos manejados en detalle, ayudaron a dar un dominio más acabado de ciertos puntos y detalles.
- El lenguaje *UML* es una herramienta poderosa de modelado, unifica el vocabulario y contempla los conceptos fundamentales del paradigma de orientación a objetos, así como los elementos de desarrollo actuales.
- El caso de estudio desarrollado fue intentado en el pasado en varias ocasiones usando metodologías tradicionales y en ningún caso fue terminado con éxito. Esto podría dar la pauta de que la metodología aporta nuevos elementos de modelado integrados de forma que facilitan la tarea del desarrollador.
- Una de las diferencias fundamentales con otras metodologías es el enfoque basado en casos de uso. El concepto de caso de uso proporciona una serie de ventajas y beneficios, con el costo de la dificultad que implica comprender el tema y el alto riesgo que representa una eventual incorrecta aplicación.
- Pueden faltar elementos cuantitativos de evaluación debido a que la metodología fue desarrollada en paralelo con el caso de estudio.

6.2 Trabajos futuros

Lo siguiente podría ser complemento para el presente trabajo:

- Consolidar la metodología con un nuevo proyecto que la aplique.
- Completar la metodología con el resto de las etapas de desarrollo.
- Completar validaciones, testeo y otros detalles de la implementación.
- Permitir el uso de métodos de optimización para la simulación de financiaciones.

Parte III

Apéndices

Apéndice A
Conceptos Usados

A.1 Introducción

En este apéndice se tratan algunos conceptos usados en el proyecto. La razón de por qué incluirlos en este documento, es que a pesar de ser independientes del *UML*, resultan básicos y dado que pueden resultar desconocidos para el lector, merecen un tratamiento especial. Para un lector familiarizado con ellos, la lectura de este apéndice es opcional. Concretamente, estos conceptos son los casos de uso y los contratos de software.

A.2 Casos de uso

Un caso de uso es una descripción narrativa de un conjunto de secuencias de acciones (incluyendo variantes) que un sistema realiza y que conduce a un resultado de valor para un actor (agente externo). Dicho en otras palabras, un caso de uso es una transcripción completa del diálogo entre un agente externo y el sistema cuando este agente usa al sistema. Este diálogo finaliza cuando el actor obtiene el resultado que esperaba. Cabe aclarar que un actor puede ser tanto una persona como otro sistema con el que nuestro sistema interactua.

La principal dificultad que presenta este enfoque para su comprensión, es llegar a una medida de qué es un "uso" en los términos que se manejaron antes. Por ejemplo, una persona que desea hacer un depósito y una transferencia de dinero puede ser considerado con un actor en el sistema del cajero automático. El resultado esperado por esa persona luego de salir del cajero es que el cajero le acepte el depósito y realice la transferencia. Por lo tanto, puede considerarse que existe un único caso de uso que es "hacer el depósito y la transferencia" o en cambio considerar que existen dos casos de uso "hacer un depósito" y otro "hacer una transferencia" y que la persona del ejemplo usó el sistema de esas dos maneras. Dado que el nivel de granularidad de la segunda opción parece ser más general, resulta preferible a la primera.
Al aplicar lo anterior, es posible incurrir en un exceso y manejar un nivel de granularidad tal que se consideran acciones simples como casos de uso. En este caso es necesario tener presente que un caso de uso es un diálogo *completo*. Por ejemplo, se podría considerar que el ingreso de la tarjeta y el PIN es un caso de uso. Excepto que quiera jugar con el cajero, una persona usualmente no ingresa su tarjeta y PIN para luego retirarse.

No existe una estructura estándar para especificar un caso de uso y distintos autores proponen la suya propia. Para este proyecto se utilizó la propuesta por Larman y se detalla en las siguientes subsecciones. Un documento que contenga un caso de uso se puede categorizar, dependiendo del nivel de detalle en casos de uso de alto nivel y casos de uso expandidos. A su vez se puede clasificar el tipo del caso de uso según su nivel de importancia en primarios, secundarios y opcionales.

A.2.1 Casos de uso de alto nivel

Los casos de uso de alto nivel describen los procesos en forma breve, usualmente en dos o tres oraciones. Resultan adecuados para una primera aproximación al estudio de los casos de uso. A continuación se muestra y explica la estructura de un documento de caso de uso de alto nivel:

Caso de uso:	**Nombre del caso de uso**
Actores:	Lista de actores que interactuan con el sistema en el caso de uso.
Tipo:	Nivel de importancia (primario, secundario u opcional).
Descripción:	Descripción breve en lenguaje natural de la interacción entre los actores y el sistema.

A.2.2 Casos de uso expandidos

Los casos de uso expandidos describen los procesos en forma más detallada que los de alto nivel. La diferencia fundamental es que contienen una sección llamada *Curso típico de eventos* en la que se describe la interacción paso a paso. Los casos de uso expandidos se usan para interiorizarse más en los detalles de un caso de uso que previamente fuera expresado en alto nivel. A continuación se muestra y explica la estructura de un documento de caso de uso expandido:

La parte superior del documento es información general.

Caso de uso:	**Nombre del caso de uso**
Actores:	Lista de actores que interactuan con el sistema, indicando cuál inicia el caso de uso.
Propósito:	Intención del caso de uso.
Sinopsis:	Repetición de la descripción del documento de alto nivel.
Tipo:	Nivel de importancia (primario, secundario u opcional).
Referencias cruzadas:	Casos de uso o requerimientos relacionados.

La parte central es el corazón del documento expandido y contiene un detalle del diálogo entre los actores y el sistema. Es importante tener en cuenta que en esta parte se describe solamente la secuencia típica de eventos en un proceso exitoso. Situaciones alternativas no se incluyen en el curso típico.

Curso Típico de Eventos

Acción del Actor	**Respuesta del Sistema**
Acciones de los actores numeradas en orden de ocurrencia.	Descripción de las respuestas del sistema numeradas en orden de ocurrencia.

La última sección del documento, *Cursos Alternativos*, describe excepciones que puedan ocurrir respecto del curso típico.

Cursos Alternativos

- Alternativa que pueda ocurrir en un determinado número de línea. Descripción de la excepción.

A.3 Contratos de software

Un contrato de software es un documento que describe qué es lo que una operación debe conseguir. Usualmente es de estilo declarativo, con un neto énfasis en *qué* es lo que debe ocurrir y no en *cómo* es que se consigue.
La esencia de un contrato es describir el estado del sistema *antes* de la ejecución de la operación en lo que se denomina *precondiciones*, y describir el estado del sistema *luego* de la ejecución de la operación en lo que se denomina *postcondiciones*. De este modo, el efecto de la ejecución de la operación del sistema es tomar al sistema en el estado especificado en las precondiciones y dejarlo en el estado especificado en las postcondiciones. Esa es la forma que se tiene de describir qué es lo que se debe hacer, sin decir cómo.
A continuación se muestra y explica la estructura de un contrato de software:

Contrato

Nombre:	Nombre de la operación para la cual se escribe el contrato.
Responsabilidades:	Responsabilidades de la operación.
Tipo:	Tipo del cual la operación es propiedad (tipo, clase o interfaz).
Referencias cruzadas:	Casos de uso o requerimientos relacionados.
Notas:	Comentarios o restricciones.
Excepciones:	Casos excepcionales.
Salida:	Salida de la operación.
Precondiciones:	Asunciones sobre el estado de la instancia antes de la ejecución de la operación.
Postcondiciones:	Estado de la instancia luego de la ejecución de la operación.

A.3.1 Postcondiciones

Las postcondiciones, junto con las responsabilidades, son la parte más importante de un contrato de software. No son acciones a realizar durante la ejecución de la operación, sino que son declaraciones acerca del estado del sistema que deben ser satisfechas luego de que la operación finaliza su ejecución. Los cambios en el estado de la instancia deben ser expresados en términos de las siguientes categorías:

- Creación y eliminación de instancias.
- Modificación de atributos.
- Links formados y rotos.

Es usual referir los cambios del sistema en términos del contenido de un modelo conceptual. Por eso, las instancias creadas son las del modelo conceptual y los links formados son relativos a las asociaciones del modelo conceptual.