PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

MAESTRÍA EN INFORMÁTICA

MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DE PLANIFICACIÓN DE CONSTRUCCIÓN DE CASOS DE USO

ELABORADO POR : Arturo Moquillaza Vizarreta
amoquillaza@pucp.pe

ASESOR : José Antonio Pow Sang

JURADOS : José Antonio Pow Sang
Juan Miguel Angel Guanira
Héctor Andrés Melgar

ÁREA DEL PROYECTO : Ingeniería de Software

TIPO DE PROYECTO : Proyecto de Implementación

Lima, julio de 2013

Contenido

Capítulo 1: Introducción

Los procesos actuales de desarrollo de *software* implican la utilización de una serie de herramientas y de metodologías que permitan dirigir y soportar el proceso de desarrollo y la gestión adecuada del proyecto mismo. Con el paso del tiempo, estas metodologías se han hecho más útiles a medida que la complejidad del software que se desarrolla ha ido en aumento. RUP, por ejemplo, es una metodología que nos guía para lograr este propósito **[1]**.

Uno de los grandes desafíos de la comunidad de investigadores en ingeniería de *software*, es involucrar a los *stakeholders* en el proceso de captura de requerimientos. Un *stakeholder* se puede definir como aquella persona que está materialmente afectada por el resultado del proyecto. En ese sentido, todo proyecto involucra la satisfacción de necesidades de un grupo diverso de *stakeholders*. Típicamente, estos tienen diferentes perspectivas sobre el problema, y diferentes necesidades que deben ser convenientemente identificadas **[4]**.

No obstante los avances logrados en los últimos años, hay algunas partes del proceso de desarrollo de *software* que aún no están adecuadamente soportadas por alguna metodología, y en ese sentido, requieren de la experiencia de quienes manejan el proyecto de *software*.

El proceso de elección del orden de construcción de los casos de uso a partir de los requerimientos capturados, es un ejemplo de lo anterior.

El presente proyecto muestra una propuesta de solución a este tema. Se ha implementado un módulo de *software* que permite capturar la opinión de los *stakeholders* y a partir de estas, generar una secuencia de construcción de casos de uso.

1.1. Estructura del documento

El presente informe se divide en ocho capítulos. El primero presenta un breve resumen sobre el problema tratado, y sobre la solución que se ha implementado sobre el mismo.

En el segundo capítulo se trata el contexto de la solución. Esto es, tecnologías, metodologías y propuestas, revisadas y tomadas para diseñar el módulo que se ha desarrollado.

En el tercer capítulo se revisa con más detalle el problema. Se desarrolla una justificación sobre el presente trabajo, se presenta los objetivos, y se describe la metodología utilizada para desarrollar el presente trabajo.

A partir de aquí, y sobre la metodología expuesta, se presenta el desarrollo del módulo. Así, en el capítulo cuarto, se muestra el proceso de captura de requisitos.

En el capítulo quinto, se muestra el proceso de análisis de la solución, y se presenta el modelo para el algoritmo que se implementó para procesar las encuestas.

En el capítulo sexto se presenta la arquitectura y el diseño de la solución desarrollada.

En el capítulo sétimo se presenta en módulo el funcionamiento, como una prueba sobre el mismo. Así, el resultado coincide con el modelo presentado en el capítulo quinto.

En el capítulo octavo, se presenta las conclusiones obtenidas del desarrollo del presente trabajo, y de la misma forma, se plantea recomendaciones para trabajos futuros.

Se tiene por último una sección de anexos donde se puede encontrar información adicional sobre los procesos de análisis y diseño.

1.2. Marco conceptual

Dentro de este apartado, se revisa algunos de los conceptos que se utilizarán en capítulos siguientes. Estos conceptos fueron de gran utilidad para entender la problemática expresada, y ayudaron a proponer una solución a la problemática planteada.

1.2.1. Planificación de entrega de Software

Software release planning o planificación de entrega de software, es una parte importante de las actividades de un desarrollo incremental de *software*.

La idea principal del desarrollo incremental de software es construir un sistema de manera creciente, permitiendo al programador sacar ventaja de lo que se ha aprendido en las etapas previas de construcción, incrementando versiones entregables del sistema de *software.* Este aprendizaje viene de dos fuentes: el desarrollo del sistema y su uso (en tanto sea posible). En cada entrega se pueden realizar cambios en el diseño y se agregan nuevas funcionalidad y capacidades al sistema en desarrollo.

Release Planning, en ese sentido, dirige el proceso de decidir cuáles requerimientos se deben asignar a cada entrega de un *software* en evolución. Es decir, es el proceso de asignar características o requisitos a las entregas de un producto. Las investigaciones que se han hecho en esa área **[4]**, **[9]**, **[10]** y **[11]**, sugieren que un planeamiento sistematizado basado en una herramienta que lo soporte, incrementa la confianza en la solución, y es más fidedigno que el planeamiento *ad-hoc*.

El objetivo principal de un *software release planning* es encontrar los más prometedores planes de entrega o *release plan* que logren la satisfacción de las partes interesadas en un proyecto, maximizando los recursos disponibles, teniendo en cuentas las limitaciones existentes.

Las decisiones de cuáles requisitos se incluirán en una entrega son complicadas, especialmente frente a un proyecto que tiene muchos requisitos y un gran número de *stakeholders*. Además se convierte en un problema aún más difícil cuando la estimación de recursos, las capacidades de recursos y dependencias entre los requisitos se toman en cuenta.

Los *release planning* tienen un impacto sobre el tiempo de comercialización de un producto, satisfacción de los usuarios y la estabilidad del proceso de desarrollo. Por ello para que un *release planning* tenga éxito se debe considerar el esfuerzo, las finanzas y los riesgos **[11].**

1.2.2. Ad Hoc Planning

Ad Hoc Planning es uno de los enfoques de un *Software Release Planning*. Este planeamiento *Ad Hoc* está principalmente basado en la intuición humana, comunicación y capacidades humanas para decidir cuáles requerimientos deben ser seleccionados en una entrega **[10]**. Es decir se basa en la experiencia de los desarrolladores y en sus opiniones de expertos.

La desventaja de esta técnica es que cada proyecto de *software* tiene diferentes características y son vulnerables a distintas situaciones, lo cual implica que no siempre se tenga éxito utilizando sólo la experiencia de los desarrolladores **[9].**

1.2.3. Systematic Planning

Este enfoque está basado en la formalización del proceso. Se asume una descripción cuantitativa del problema y la aplicación de algoritmos de optimización para su solución **[10]**. Esto involucra una serie de parámetros para la adaptación del problema a un contexto determinado.

Systematic Planning es un enfoque mucho más completo, pues no solo se puede tomar en cuenta la experiencia de los desarrolladores, sino que también hace uso de algoritmos adecuados para la evaluación de las entregas, los cuales optimizan los resultados.

Ruhe en **[4], [9]**, **[10]** y **[11]**, da cuenta de sus investigaciones sobre el tema. Llegó incluso a formalizar un método para capturar la opinión de los *stakeholders* sobre los requerimientos identificados, y a plasmar todo este conocimiento en una herramienta de soporte llamada *ReleasePlanner*.

1.2.4. StarUML - The Open Source UML/MDA Platform

StarUML es una herramienta *open source* para desarrollar proyectos de *software* de manera rápida, flexible y extensible, bajo la plataforma UML/MDA[1] sobre ambientes Win32 **[3].** Esta herramienta se basa en estos pilares: UML 2.0, MDA, Arquitectura basada en *Plug-in* y Usabilidad.

StarUML está escrito casi en su totalidad en el lenguaje *Delphi*. No obstante, *StarUML* es un proyecto multilenguaje; además, *StarUML* provee una arquitectura basada en *plug-ins*, que permiten añadir nuevas funcionalidades a manera de módulos (*plug-ins*) que interactúan con su API mediante interfaces estándar. Esto permite que muchos otros lenguajes puedan usarse para seguir desarrollando *StarUML* (por ejemplo, C/C++, Java, Delphi, C#, VB.NET, etc.).

La información referente a la propia API de *StarUML* para interactuar con la aplicación vía objetos COM, se extrae de la propia página del proyecto StarUML[2]. La figura 1 muestra información de la API de StarUML.

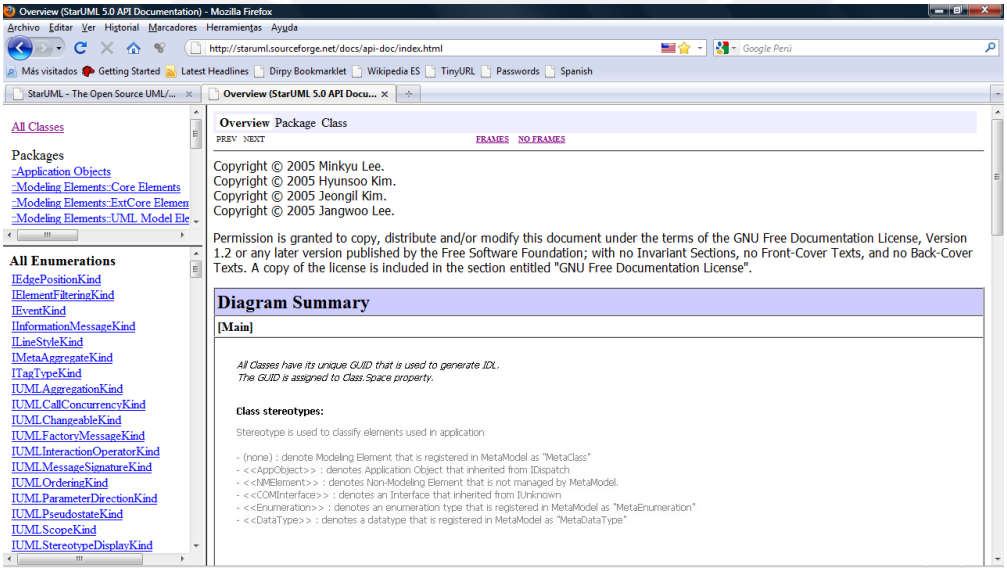

Figura 1: Información de la API de StarUML (captura de pantalla)

[1] *UML son las siglas de Unified Modeling Language o Lenguaje Unificado de Modelado y MDA son las siglas de Model Driven Architecture o Arquitectura dirigida por el modelo.*

[2] http://staruml.sourceforge.net/docs/api-doc/index.html

El crecimiento y enriquecimiento de la funcionalidad de *StarUML* se realiza mediante su arquitectura basada en *plug-ins*. Esto significa interfaces de programación estándar que interactúan con las librerías API de *StarUML*.

1.2.5. UML (Unified Modeling Language)

UML es un lenguaje estándar de modelado utilizado para diseñar, especificar, construir y documentar artefactos de sistemas de software **[1]**.

UML es un lenguaje gráfico usado para visualizar, especificar, construir y documentar un sistema. Este lenguaje ofrece un estándar para describir un modelo del sistema, lo que incluye tanto aspectos conceptuales (como procesos del negocio), funciones propias del sistema y aspectos concretos como expresiones propias de leguajes de programación, esquemas de bases de datos, etc.

UML está en continua expansión y es manejado por la OMG (*Object Management Group*). En la actualidad (2012) la versión más reciente es UML 2.0.

1.2.6. MDA (Model Driven Architecture)

MDA es una nueva tecnología introducida por la OMG[3].

MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos. Uno de los principales objetivos de MDA es separar el diseño de la arquitectura y de las tecnologías que se puedan emplear para la construcción del sistema, lo que facilita que el diseño y la arquitectura puedan ser modificados de manera independiente. En ese sentido, el diseño contiene los requerimientos funcionales (mediante los casos de uso por ejemplo) mientras que la arquitectura proporciona la infraestructura mediante de la cual se hacen efectivos requerimientos no funcionales como escalabilidad, fiabilidad o rendimiento.

Para sacar todo el provecho posible de MDA, la herramienta de modelado de *software* requiere soportar muchas variables personalizadas. Con base en los estándares establecidos por OMG, MDA separa el negocio y la lógica de aplicación de la plataforma tecnológica. MDA es soportado por *StarUML*.

1.3. Propuesta de solución

En los proyectos de desarrollo de *software*, una de las técnicas más utilizadas es la de Casos de Uso para definir los requerimientos del *software*. Estos casos de uso son necesarios, entre otras cosas, para determinar la secuencia de cómo el *software* será construido.

Una vez que se ha obtenido los requerimientos del *software* que se va a construir, es responsabilidad de quien lidera el proyecto determinar qué se construirá primero. En ese sentido, se negocia o se acuerda con el cliente el orden de desarrollo, de acuerdo a factores internos o externos que afectarán el desarrollo del proyecto. Para ello tenemos dos fuentes: las restricciones propias del proyecto, que son considerados factores externos, y la opinión de los desarrolladores, los cuales basan la misma en su experiencia con otros proyectos, para determinar de esta forma la mejor alternativa que satisfaga las restricciones del proyecto.

Por otro lado, la herramienta *StarUML* **[3]**, que usa el lenguaje UML para modelar el desarrollo de *software*, es usada por los desarrolladores de un proyecto como apoyo en las actividades

[3] http://www.omg.org/mda/

que surgen en la construcción de *software*. Pero los desarrolladores tienen que usar distintas herramientas que los apoyen en los procesos de *software* dependiendo de las funcionalidades que necesiten, lo cual retrasa y hace complejo el desarrollo.

Según lo anterior, se ha analizado, diseñado y construido un módulo (*plug-in*) para la herramienta *StarUML* que permite generar una matriz de orden de construcción de casos de uso a partir de un diagrama de secuencias y de votaciones de los involucrados en el proyecto de desarrollo.

Para lograr lo anterior, se elaboró el análisis y el diseño del módulo propuesto, y luego se procedió a la construcción del mismo, dando como resultado final un módulo ejecutable.

Capítulo 2: Contexto del Problema

En este apartado, se muestran las herramientas y propuestas existentes revisadas para plantear una propuesta de solución. Para ello se describen las metodologías y técnicas usadas por estas aplicaciones, dando a conocer cómo funcionan y cuáles son los criterios básicos que se utilizan.

2.1 Técnica para determinar la secuencia de construcción de requisitos de Software basados en Casos de Uso

El diagrama de precedencia de casos de uso (*Use Case Precedence Diagram* - UCPD) es un diagrama de casos de uso que propone la inclusión de una nueva relación: *precedencia* **[2].** Esta relación define qué casos de uso se deben realizar primero para poder realizar otros, es decir determina la secuencia en las cuales se ejecutaran los casos de uso, y esto ayudara a saber qué secuencia de construcción se debe seguir en un proyecto de *software*. El concepto de este diagrama se toma del trabajo de Doug Rosenberg **[12]**, quien propone usar relaciones del tipo "precede" e "invoca" para determinar requerimientos de usuario. No obstante, es en **[2]** donde se propone su utilización para establecer la secuencia de construcción, como un método más formal que el utilizado sólo como base en la experiencia de los desarrolladores.

Un ejemplo de este diagrama se muestra en la figura 2.

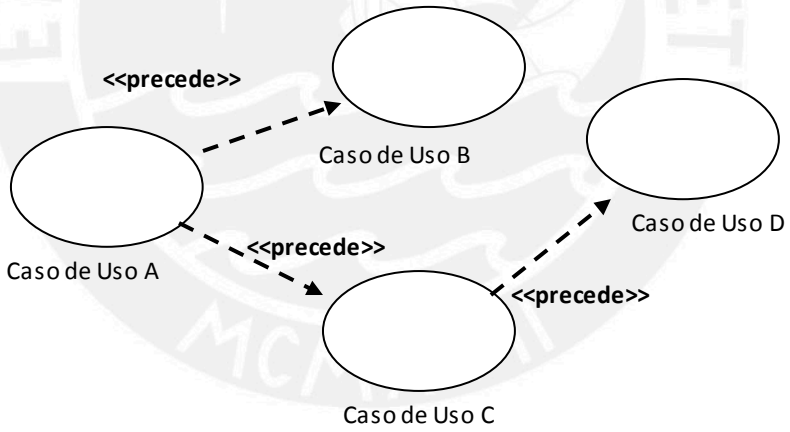

Figura 2. Diagrama de Precedencia de Casos de Uso

Para entender las relaciones de precedencia se debe tener en cuenta dos reglas:

Regla 1: Un caso de uso U1 precede a otro caso de uso U2 si hay una precondición que corresponde a la ejecución de un escenario en U1 que debe ser ejecutado antes de un escenario U2. Vemos en la figura 3 un ejemplo de lo expuesto:

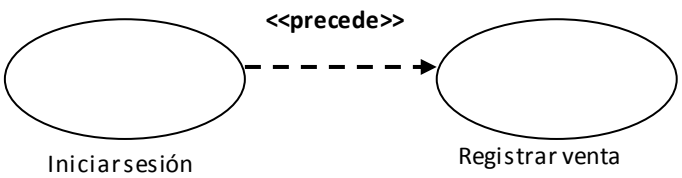

Figura 3. Regla 1 de Precedencia

Regla 2: Un caso de uso U1 precede a otro caso de uso U2 si U2 necesita información que es registrada por el caso de uso U1. Por ejemplo para llevar a cabo el pago de una reservación, esta reservación debe haberse ingresado, como se muestra en la figura 4.

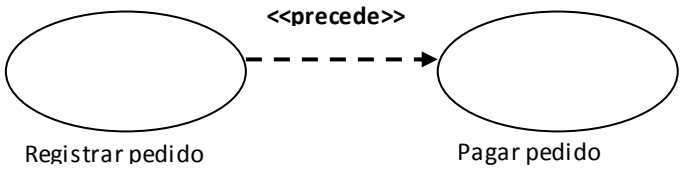

Figura 4. Regla 2 de Precedencia

Así, este diagrama de precedencias de casos de uso, ayuda a definir la construcción de secuencias que tendrá un proyecto de software.

Con base en este diagrama de precedencia de casos de uso, se puede definir una secuencia de construcción: el orden de implementación consistirá en construir los casos de uso de izquierda a derecha.

A medida que la cantidad de casos de uso aumenta, este diagrama se hace más grande, en ese sentido, la manipulación y lectura de este diagrama tiene a hacerse más complejo. TUPUX visto en el apartado 2.3 implementa la forma de graficar dentro de la herramienta *StarUML* estos diagramas, pero queda pendiente la forma de leerlos de una manera más ágil y resumida.

2.2. Herramienta para el Diseño de la Estructura Matriz (DSM)

Siempre en cualquier tipo de negocio está presente, como un obstáculo importante, la complejidad. Por ello se debe conocer y utilizar una gestión de sistemas complejos, para alcanzar el éxito en un negocio de cualquier índole.

Como respuesta para afrontar la complejidad presente en cualquier negocio, nace una herramienta llamada *The Design Structure Matrix* o Diseño de la Estructura Matriz (DSM) cuyo objetivo es realizar tanto el análisis y la gestión de sistemas complejos **[6].**

Entre sus principales características podemos mencionar que permite al usuario modelar, visualizar y analizar las dependencias entre las entidades de cualquier sistema y obtener sugerencias para la mejora o la síntesis de un sistema, evitando así la complejidad en el desarrollo.

Así tenemos que DSM ayuda a minimizar la complejidad de un sistema, pero en qué tipos de sistemas se puede aplicar dicha herramienta; como respuesta tenemos una gama amplia de ámbito de aplicación. Por ejemplo dichos sistemas pueden ser: la arquitectura de un producto o un proceso de diseño de ingeniería, así como también, la organización de una empresa o un mercado que puede tomar forma como un sistema complejo y, a menudo, merecen una mirada más cercana en su estructura.

Como una herramienta para análisis de sistemas, DSM ofrece un compacto y una clara representación de un complejo sistema y un método para capturar las interacciones, interdependencias, interfaces entre elementos del sistema, es decir, subsistemas y módulos.

Como un instrumento de gestión, DSM más comúnmente aplicado en la gestión de proyectos, proporciona la representación de un proyecto que permite la retroalimentación y dependencia de tareas cíclicas. Esto es sumamente importante ya que la mayoría de aplicaciones de ingeniería tienen como propiedad ser cíclicos. Por lo tanto, esta representación a menudo resulta una mejor y más realista planificación de ejecución para el diseño de actividades.

El diseño de una estructura matriz, como su nombre lo indica, es básicamente una matriz usada para relacionar entidades de un tipo con cada una de otros tipos. Para entender esta definición, pensemos en las tareas que constituyen un proyecto completo y que puede ser usado para identificar apropiados equipos de trabajo y una secuencia ideal y efectiva de cómo pueden las tareas ser organizadas. En la figura 5 se muestra ejemplos de estas matrices.

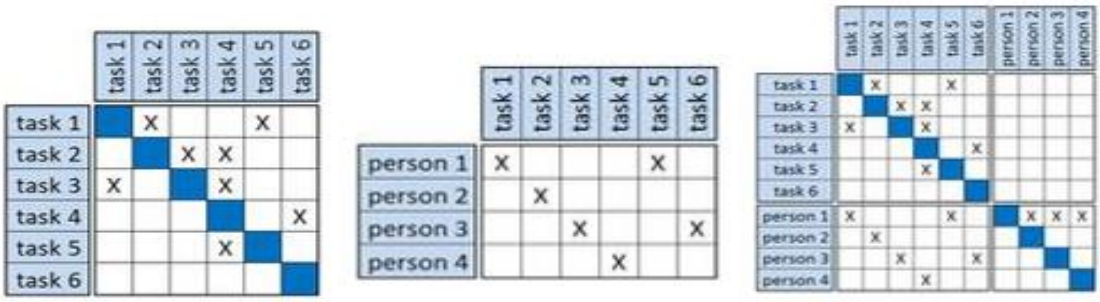

Figura 5. Ejemplo de las matrices generadas por DSM (tomado de [6])

El DSM es capaz de modelar y analizar las dependencias de un solo tipo en un único dominio. Para un producto, por ejemplo, el dominio de los "componentes" se puede considerar. Utilizando el tipo de relación "cambio de componente 1 causa el cambio de componente 2", un conjunto se puede analizar con respecto a los impactos del cambio global, con el fin de modelar las posibles propagaciones de cambio.
Así vemos que esta herramienta nos da una solución al desarrollar un sistema complejo, permitiendo analizar y minimizar la complejidad de dicho sistema para un correcto desarrollo e implementación de este.

La utilización de estas matrices nos permite también identificar todos los dominios o elementos del problema, establecer su interrelación, y mediante la aplicación de los algoritmos propios de DSM, nos permite optimizar el modelo, añadir o fusionar dominios del problema, partir el problema en otros más pequeños, etc.

2.3. TUPUX

TUPUX es una herramienta de estimación para el desarrollo de proyectos de software incrementales **[7]**. Esta herramienta fue desarrollada como soporte en el proceso de estimación de esfuerzos en un proyecto de *software*. Este módulo es un componente que se agrega a la herramienta *StarUML*, el cual facilita la planificación y estimación, usando Puntos de Función, en proyectos de *software* que tengan un modelo incremental.

Además TUPUX soporta especificaciones de casos de uso como una actividad previa a la estimación de esfuerzos.

La técnica usada para el planeamiento y estimación de desarrollo de *software* incremental consta de dos fases. La primera determina los casos de uso que serán desarrollados en cada incremento, determinando la secuencia de construcción de casos de uso. La segunda fase contiene la estimación de esfuerzos por cada incremento a desarrollar.

La primera fase tiene como objetivo determinar las secuencias de construcción de los casos de uso, para ello se determina el diagrama de precedencias de casos de uso el cual será utilizado para determinar el orden lógico de la construcción de casos de uso.

En la segunda fase se determina el esfuerzo por incremento. Lo cual se logra aplicando varias formulas de solución.

2.4. ReleasePlanner

ReleasePlanner es una herramienta cuyo objetivo es mitigar los problemas de un proceso de planeamiento *Ad-hoc*, dando la opción de un enfoque más sistemático **[11]**.
Como ya se comentó, recoge el conocimiento y las recomendaciones de Ruhe, sobre sus investigaciones en *Release Planning*.
Esta herramienta apoya en todas las actividades que se necesitan en un proceso de planeamiento. En la Figura 6 se muestra un *workflow* del proceso de planeamiento que utiliza *ReleasePlanner*.

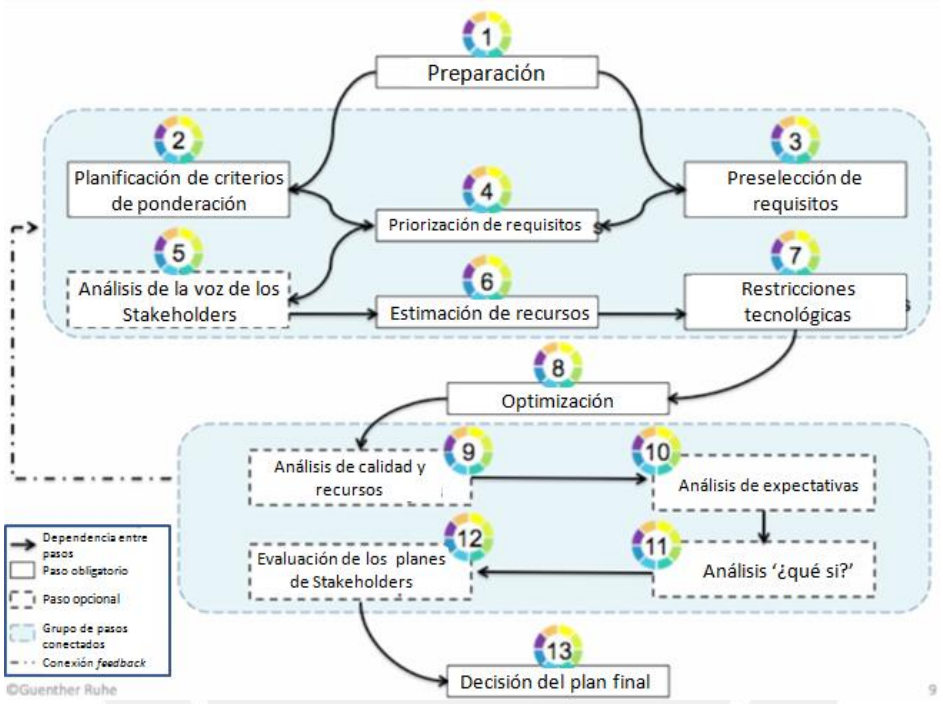

Figura 6. Modelo del Proceso del plan de entrega

La información y capturas a continuación ha sido obtenida de los videos de ayuda en línea (tutorial series) [4]
ReleasePlanner tiene tres grupos de actividades bien definidas:

- Recolección e importación de datos del proyecto
- Votación de los *Stakeholders* y análisis
- Generación y análisis de los planes optimizados

Se indica a continuación algunos lineamientos de cada uno de estos grupos de actividades:

a) Recolección e importación de datos del proyecto

La colección de datos del *release planning* se inicia especificando los parámetros clave del planeamiento, los cuales son:

- Objetos de planeamiento (características, requerimientos servicios, etc.)
- Número de entregas
- Criterios de planeamiento
- Número de *stakeholders* del proyecto
- Número de recursos considerados en el proyecto

[4] *Disponibles en https://www.releaseplanner.com/RP_Help_Tutorial_Series.htm*

Para llevar a cabo este proceso, primero se genera una hoja MS Excel con los parámetros clave a partir de una plantilla. Luego en MS Excel se termina de completar la información requerida. Por último, se carga el documento MS Excel con la información recolectada, nuevamente en la herramienta.
En resumen, mediante este grupo de actividades, se consigue:

- Establecer cuál es la información clave del proyecto necesaria para la generación del proyecto
- Ingresar la información del proyecto mediante una plantilla MS Excel predefinida.
- Importar la información del proyecto al *ReleasePlanner*
- Generar planes que se pueden exportar al MS Excel nuevamente, luego de que los *stakeholders* ya votaron y se ha ejecutado el proceso de planeamiento.

b) Votación de los Stakeholders y análisis

El flujo se inicia cuando al *Stakeholder* le llega un email personalizado con un link a la herramienta, para que pueda realizar las priorizaciones de las características y requerimientos, según su visión del proyecto.
Los criterios más usados para el planeamiento, propuestos por defecto por *ReleasePlanner* son: Urgencia,[5] Valor, Riesgo, Tiempo en el mercado y Volatilidad.
Además, existe la posibilidad de añadir más criterios personalizados.

El *stakeholder* entonces procede a realizar su votación en la herramienta. Una vez ha terminado este proceso para los demás stakeholder, el *Project manager* puede acceder a la información de las votaciones realizadas. Donde se pueden realizar análisis de las opiniones recogidas. Este proceso se llama *voice of stakeholders analysis*. Los stakeholders juegan un rol clave en el proceso de evaluación previa, priorización, estimación y generación de la decisión final. Basados en su rol en la organización y en el proceso real de desarrollo de productos, sus roles en el proceso de planeamiento necesitan ser explícitamente especificados. En la figura 7 se muestra una captura del proceso de análisis de la voz de los *stakeholders*.

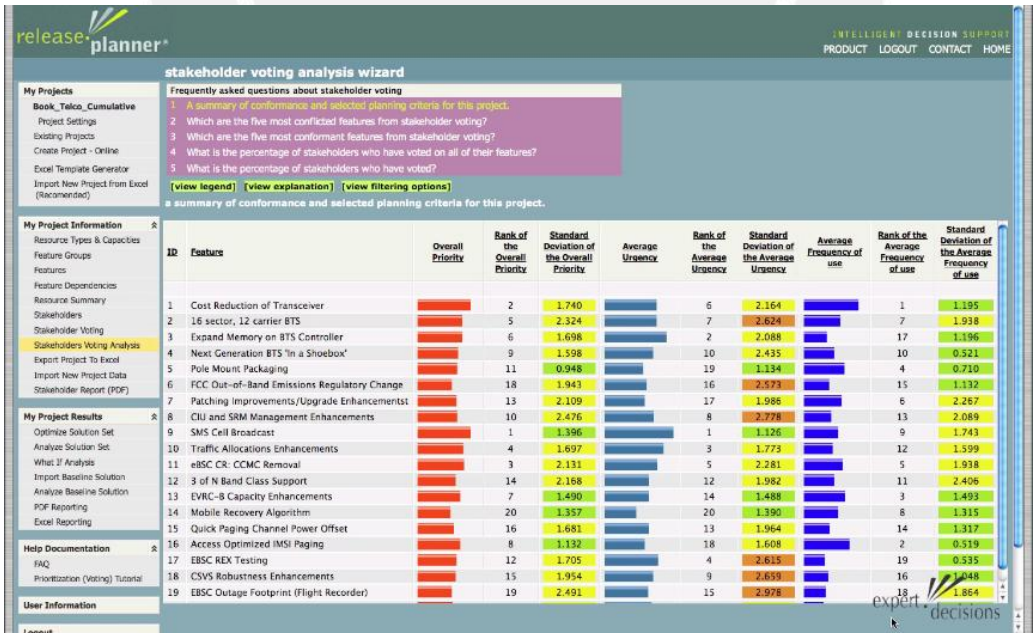

Figura 7. Análisis de la voz de los Stakeholder (captura)

[5] *En ReleasePlanner, se entiende por urgencia como el conjunto de expectativas en términos de tener esta característica entregada como parte de la siguiente entrega.*

Esta herramienta parte del principio de que al permitir acceso a los *stakeholders* de donde sea, en cualquier momento, no sólo provee más flexibilidad en el proceso real de priorización, sino que proveer mayor objetividad en general.

c) Generación y análisis de los planes optimizados

Después de que la votación de los *stakeholders* ha terminado, se puede proceder a generar un conjunto de soluciones optimizadas. Se pueden generar varias alternativas de soluciones optimizadas. La figura 8 muestra este proceso.

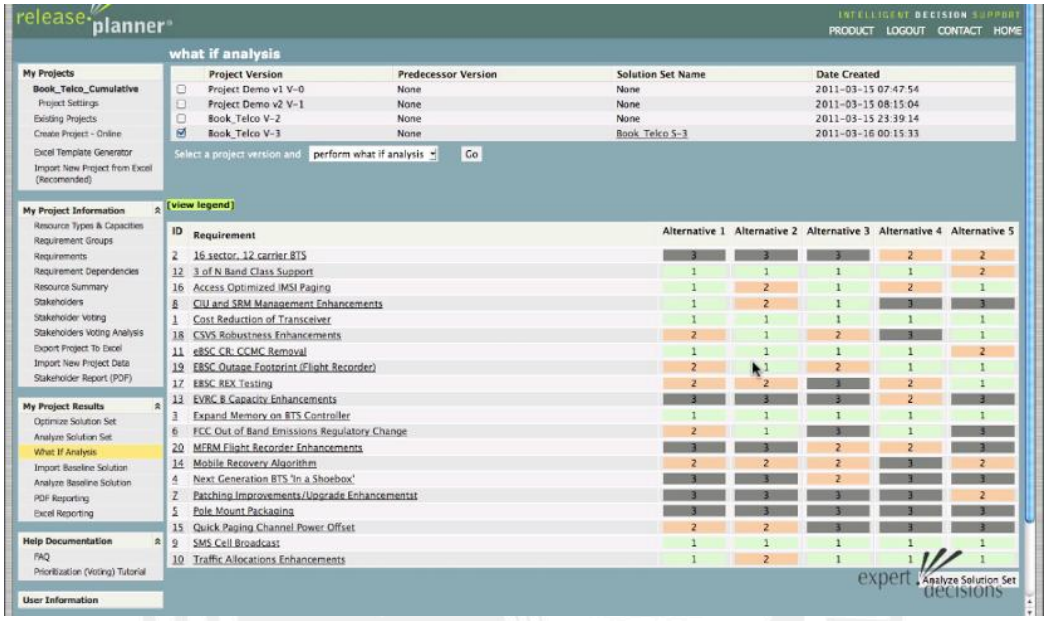

Figura 8. Generación de planes optimizados (captura)

La generación de planes y su análisis proactivo son diseñados para proveer soporte al *project manager* para realizar su decisión final basada en su propia capacidad humana. En ese sentido, la herramienta es un facilitador para que expertos humanos tomen decisiones de calidad.

En conclusión, *ReleasePlanner* es una herramienta que apoya todas las actividades relacionadas con la generación de planes de entrega, los cuales son generados tomando en cuenta aspectos importantes del proyecto y la opinión de los *stakeholders*.

2.5. Aportes a la solución

Con base en el diagrama de precedencia de casos de uso, se puede definir una secuencia de construcción: el orden de implementación consistirá en construir los casos de uso de izquierda a derecha.

En ese sentido, este será el orden de construcción primario. Se tiene que resolver el problema de capturar la opinión del *stakeholder*, y luego cruzar esa información con la que nos brinda el diagrama de precedencia de casos de uso.

Luego de revisar DSM, hemos tomado las sugerencias y lineamientos del mismo sobre cómo interactuar con los *stakeholders* para obtener los elementos del problema, para elaborar nuestras propias encuestas.

De la misma forma, ReleasePlanner no está directamente relacionada con la construcción de casos de uso, pero se tomó como base para el proyecto, ya que tiene características similares a las que tiene el modulo de planificación de construcción de casos de uso que se implementó en el presente proyecto.

Así mismo, se ha hecho uso de algunos lineamientos utilizados por ReleasePlanner presentados aquí para capturar la voz de los *stakeholders*. Además, de la idea de permitir a cualquier *stakeholder* llenar la encuesta fuera de la herramienta, y poder cargar la misma posteriormente en el *StarUML*.

El TUPUX, por último, sirve como base para la realización del presente trabajo de tesis, ya que el módulo que se implementó es un *plug-in* semejante a dicha herramienta.

Por ello, se reutilizó como base el código desarrollado de esta herramienta, ya que se hizo uso intensivo de varias de las clases base desarrolladas en esta herramienta. En ese sentido, se ha procedido a la extensión y crecimiento de este *plug-in*.

Capítulo 3: Planteamiento del problema

En el presente capítulo, se muestra la justificación del presente trabajo, y los objetivos del mismo. Se debe plantear que cualquier alternativa de solución no será un aporte definitivo, sino un inicio para la resolución del problema de definir la secuencia de construcción de los casos de uso de un sistema de *software*; debido a la complejidad inherente al proceso de construcción de *software* **[5]**; la cual viene dada tanto por la complejidad del dominio del problema, por la dificultad de gestionar el proceso de desarrollo, como por el detalle que se puede alcanzar a través del *software*, entre otros.

3.1. Justificación

Como se mencionó en apartados anteriores, se ha sugerido una nueva técnica que permite determinar la secuencia de construcción de *software* basada en diagramas de precedencia de casos de uso y en el punto de vista de los desarrolladores respecto de la prioridad de los requerimientos **[2]**. No obstante, no existe una herramienta que soporte esta metodología, y que además pueda capturar la opinión de *stakeholders* que no sean desarrolladores.

Es el caso que la literatura revisada, (**[4]** y **[11]**) sobre todo los trabajos de Ruhe al respecto, muestran que una de las formas de capturar y sistematizar la opinión de los *stakeholders* son las encuestas. En ese sentido, se hace necesaria la implementación de una herramienta que donde no sólo se pueda modelar la precedencia, sino que también se puedan asociar preguntas a los casos de uso, para después aplicar encuestas a los usuarios involucrados.

El módulo desarrollado nace como respuesta a la necesidad de contar con una herramienta que soporte los métodos propuestos por Pow-Sang **[2]** vistos en el apartado 2.1.

Dichos métodos propuestos utilizan diagramas de precedencia de casos de uso, y además proponen obtener la opinión de los *stakeholders* involucrados con el proyecto. Con esta información, establecen una referencia para obtener una secuencia de construcción de casos de uso.

No obstante, estas encuestas deben prepararse fuera de la herramienta que contiene el diagrama de precedencia de casos de uso (en este caso *StarUML*), la aplicación y sistematización de las encuestas están fuera de la misma herramienta, y la generación de la secuencia de construcción a partir de la información anterior debe hacer de forma manual.

Por ello se decidió automatizar la generación de tal diagrama de secuencia de construcción mediante dos puntos de vista: de las personas involucradas en un proyecto y el diagrama de precedencia, los cuales se interpretan y pesan mediante un algoritmo adecuado.

3.2. Objetivo General

El presente trabajo tiene como objetivo analizar, diseñar y construir un módulo para el modelador *StarUML*, que permita definir la secuencia de construcción de casos de uso para un proyecto de *software*.

3.3. Objetivos Específicos

3.3.1. Estudiar el API que tiene la herramienta *StarUML*, para extender la funcionalidad de esta herramienta.

3.3.2. Elaborar el análisis y diseño del módulo que cubra las funcionalidades de construcción de casos de uso.

3.3.3. Desarrollar el módulo bajo una metodología dirigida por casos de uso.

3.4. Metodología

Para realizar el presente trabajo, se utilizó las siguientes metodologías tanto para del proyecto como para la elaboración del módulo propuesto.

3.4.1. Metodología para la Gestión del Proyecto

La Gestión de Proyectos es la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer los requisitos del mismo.

Para el presente proyecto de tesis se aplicó una metodología básica de gestión de proyectos, mediante la aplicación e integración de procesos básicos de inicio, planificación, ejecución, seguimiento y control, y cierre.

Es importante resaltar y tener en cuenta que muchos de los procesos incluidos en la gestión de proyectos son repetitivos debido a la necesidad de elaborar gradualmente el proyecto durante su ciclo de vida.

3.4.2. Metodología aplicada para el desarrollo de la solución

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de aplicaciones de software.

Una de las metodologías de desarrollo existentes es RUP (*Rational Unified Process*). Se trata de un proceso de desarrollo de *software* y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso.

Debido a la extensión y la calidad que requiere el presente proyecto, es indispensable adoptar una metodología de desarrollo adecuada que asegure la producción de *software* de calidad que satisfaga las necesidades de los usuarios.

En ese sentido, se opta por adoptar la metodología propuesta por Jaaksi: El método simplificado **[14]**.

El trabajo de Jaaksi propone una metodología simple y práctica para desarrollar aplicaciones orientadas a objetos. Esta propuesta se basa en dos principios fundamentales: los objetos y su cooperación. Este método incluye dos notaciones y cinco fases claramente establecidas:

Notaciones:

- Diagramas de clases
- Diagramas de secuencias

Fases:

- Captura de requisitos
- Análisis
- Diseño
- Programación
- Pruebas

La figura 9 resume gráficamente el método simplificado.

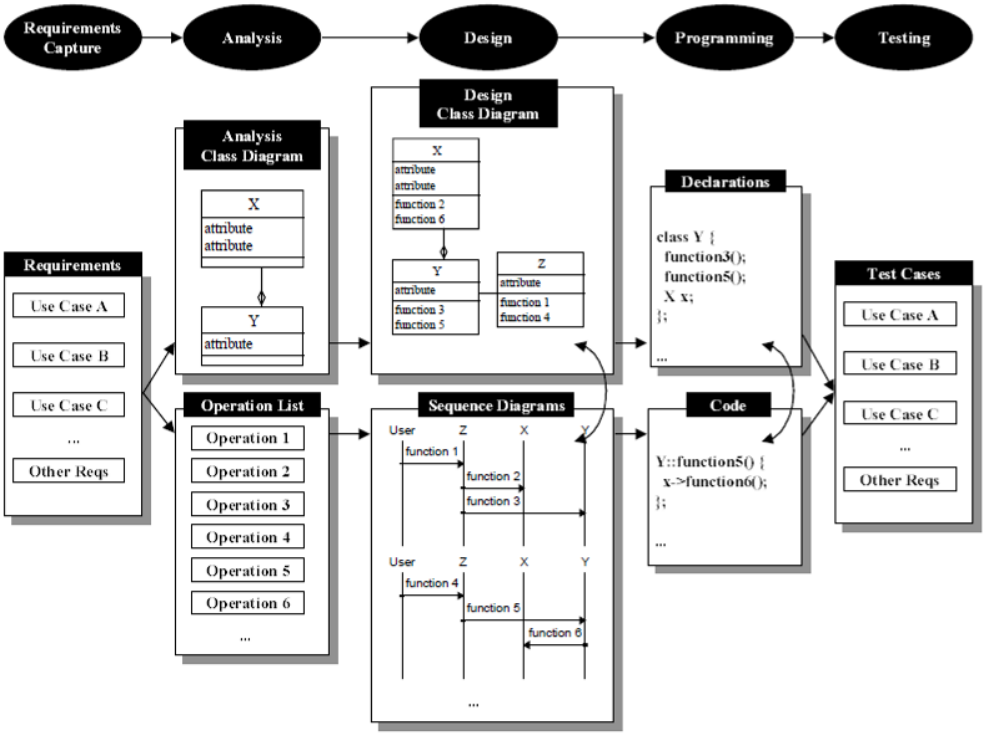

Figura 9: Método simplificado (tomado de [14])

Con base en el método simplificado, y haciendo uso de los diagramas proporcionados por UML y RUP, se ha seguido la metodología presentada en la tabla 1:

Fase	Entregables generados
Captura de requisitos	- Catálogo de requerimientos funcionales y no funcionales - Diagramas de Casos de Uso
Análisis	- Diagrama de clases de análisis - Lista de operaciones
Diseño	- Diagramas de clases de diseño - Diagramas de secuencias - Diagramas de colaboración - Pantallas de prototipo
Construcción	- Código generado - Pantallas de prototipo en funcionalidad completa
Pruebas	- Casos de pruebas - Evidencias de las pruebas

Tabla 1: Metodología utilizada (Método simplificado con C.U.)

Capítulo 4: Captura de requisitos

A continuación, se hace un repaso sobre el proceso seguido para la captura de requisitos. Con esto, se inicia el trabajo de desarrollo, según los lineamientos establecidos en la metodología propuesta.

4.1. Identificación de Requerimientos

Para la identificación y captura de requerimientos o requisitos, se toma las mejores prácticas sugeridas por Pow-Sang **[13]** y de esta manera, se establece la forma en la que se capturan los requisitos y se elaboran los casos de uso.

Los requerimientos están basados de acuerdo al alcance del presente proyecto, los cuales fueron obtenidos mediante el estudio de herramientas y metodologías que se usaron como base para el desarrollo del proyecto.

Los requerimientos obtenidos, pueden ser revisados en el catálogo de requisitos en los anexos de este documento.

Cabe mencionar que la presente especificación se ha ceñido al estándar "*IEEE Recomended Practice for Software Requeriments Specification IEEE Std 830 -1998*" **[15]**.

4.1.1. Ámbito del Sistema

El módulo es un desarrollo basado en el módulo denominado TUPUX. El motor que impulsa el desarrollo del sistema es el de implementar una herramienta que soporte la generación de una secuencia de construcción de casos de uso, basado en el diagrama de secuencias de casos de uso y en la votación de los *stakeholders*. La herramienta permite el almacenamiento de dicha secuencia, así como la creación y el llenado de las encuestas por parte de los *stakeholders* involucrados, tanto dentro del mismo módulo, como cargar las mismas, ya completadas desde alguna aplicación externa.

4.1.2. Suposiciones y Dependencias

Se asume que los requisitos descritos en este documento son estables. El módulo funciona como un *plug-in* añadido al *StarUML*. Este módulo funciona de manera *standalone* y tendrá acceso a toda la información que maneja el *StarUML*.

4.1.3. Definiciones

Project

Se refiere al proyecto generado en *StarUML* que contiene un diagrama de precedencia de casos de uso y al que se le añadirá una encuesta y luego una secuencia de construcción.

Question

Se refiere a la pregunta que se deberá asociar a una encuesta dada.

Inquest

Se refiere al grupo de preguntas orientada a capturar la opinión de los *stakeholders* y de esta manera alimentar al módulo para que pueda generar la secuencia de construcción. Esta encuesta se asociará a un proyecto en la relación de 1 a 1.

Answer

Cuando un *stakeholder* llena una encuesta, y la registra, lo que está haciendo en realidad es grabar su conjunto de respuestas. Este conjunto es la entrada para generar la secuencia de construcción.

BuildSequence

Se refiere al resultado del cálculo de las respuestas suministradas por los *stakeholders* y la valoración del diagrama de precedencia de casos de uso de un proyecto dado.

4.1.4. Modelo de Casos de Uso

A continuación se presenta los diagramas de casos de uso del sistema obtenidos durante el proceso de especificación de requisitos, los cuales permiten mostrar a alto nivel las funcionalidades que el sistema realizará. Inicialmente, se indica el catalogo de actores que interactúa con el sistema y posteriormente la descripción de cada uno de los paquetes con sus respectivos diagramas de casos de uso. La figura 10 muestra el catálogo de actores.

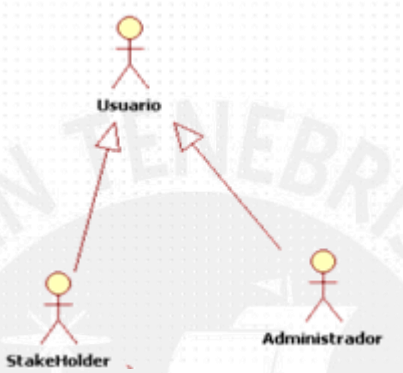

Figura 10: Catálogo de actores

Stakeholder
Representa a cualquier persona que trabajará con el sistema.
Se refiere a cualquier involucrado directa o indirectamente con el proyecto al que se le pide complete la encuesta asociada a un proyecto de interés para el mismo.

Administrador
Representa a cualquier persona que trabajará con el sistema.
Se refiere al usuario con permiso para crear y asociar una encuesta a un proyecto dado, así como generar la secuencia de construcción de casos de uso. Está capacitado para interactuar con cualquier funcionalidad del módulo.

Casos de Uso
La figura 11 muestra el diagrama de casos de uso.

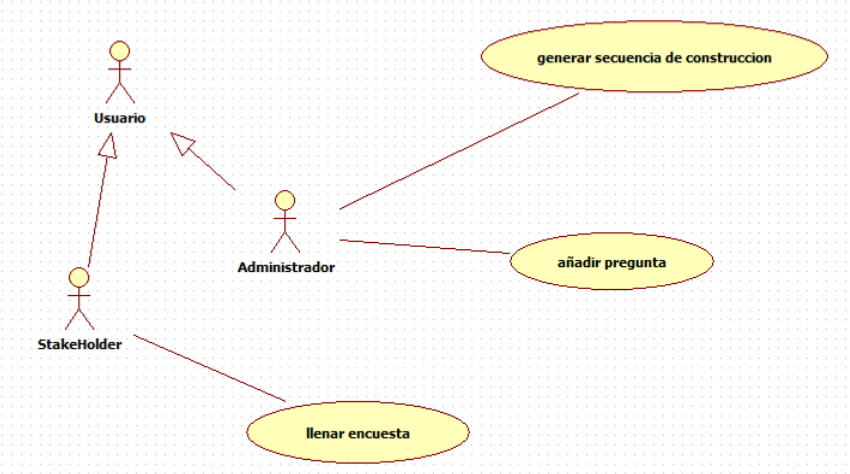

Figura 11: Diagrama de Casos de Uso

Añadir pregunta
El propósito de este caso de uso es permitir al administrador del módulo crear y asociar preguntas (Question) a cada Caso de Uso de un proyecto dado. Esta asociación permite que dicha encuesta esté disponible para que cualquier *stakeholder* pueda llenar una encuesta (Inquest), la cual es el conjunto de todas las preguntas añadidas.

Llenar encuesta
El propósito de este caso de uso es permitir al *stakeholder* recuperar el conjunto de preguntas de un proyecto (Inquest) y proceder a llenar sus respuestas en la misma. El *stakeholder* guardará sus respuestas y con esto permitirá generar la información necesaria para la generación de la secuencia de construcción.

Generar secuencia de construcción
El propósito de este caso de uso es permitir al administrador del módulo generar la secuencia de construcción de casos de uso a partir de la información recibida de dos fuentes: Dado un proyecto, el diagrama de precedencia de casos de uso asociado al mismo, y el conjunto de respuestas que los *stakeholders* guardaron al llenar la encuesta asociada al proyecto.

4.1.5. Características de los Usuarios

El módulo debe ofrecer una interfaz de usuario orientada a ventanas y debe correr en computadoras personales Pentium IV y superior. Debe integrarse a la herramienta de modelado *StarUML* de manera estándar vía su interfaz de *plug-ins* y, debe ser invocada desde la herramienta mencionada.

Se deduce de los usuarios del sistema, que los administradores serán personas con altos conocimiento en Informática y con experiencia en el manejo de aplicaciones de modelado a los que les será fácil la utilización de este módulo. Y que los *stakeholders* son personas con conocimientos intermedios de informática, y que por tanto, las encuestas deben ser lo suficientemente sencillas de llenar para estos usuarios.

4.2. Diagrama de precedencias de Casos de Uso

El diagrama de precedencias para este grupo de casos de uso ya se puede esbozar en la figura 12.

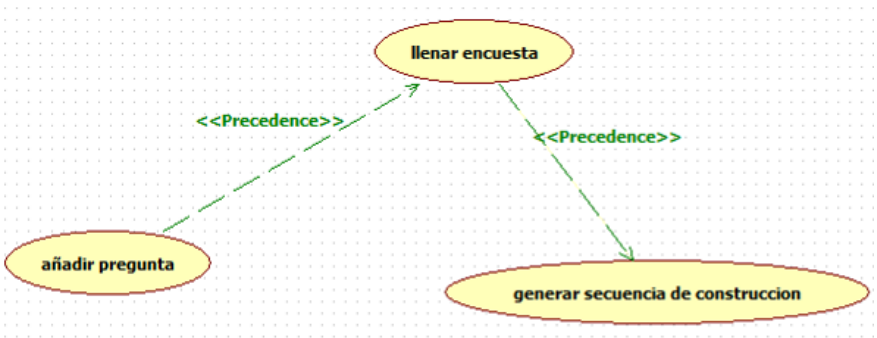

Figura 12: Diagrama de precedencia de casos de uso

Capítulo 5: Análisis

A continuación, se muestra el análisis realizado para el presente desarrollo. También se presenta el modelo sobre el cual se desarrolla el algoritmo utilizado para procesar las encuestas.

En ese sentido, se prosigue con la fase de análisis del proceso de desarrollo de *software* iniciado, según la metodología de desarrollo a la que se ha alineado este proyecto.

5.1. Clases de Análisis y Modelo Conceptual

Para proseguir con la segunda fase de la metodología de desarrollo de software adoptada (el método simplificado adaptado), se realiza el análisis del módulo.

Esta sección muestra las clases de análisis y el modelo conceptual del sistema. Los principales atributos de cada clase de análisis, pueden revisarse en la sección de anexos del presente documento.

5.1.1. Diagrama de clases de análisis (Modelo conceptual)

En la figura 13 se muestra el modelo conceptual.

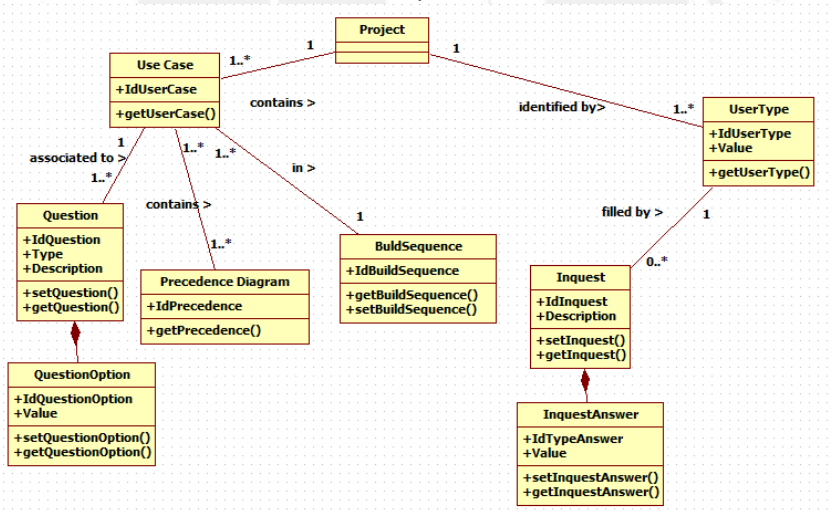

Figura 13. Modelo conceptual

5.1.2. Clases de análisis

Project

Se refiere al proyecto de *StarUML* sobre el que se han modelado los casos de uso de los que se desea obtener su secuencia de construcción.

Question

Se refiere a las preguntas que han sido asociadas a los casos de uso de un proyecto.

QuestionOption

Se refiere a las opciones por cada pregunta creada. Cada Question tiene cinco QuestionOption, que tienen un valor parametrizable y un texto modificable.

Inquest

Se refiere al conjunto de preguntas asociadas a los casos de uso del proyecto, ya respondidas. Es decir, es el contenedor de respuestas, único por cada encuesta llena.

InquestAnswer

El acto de llenar y guardar una encuesta por parte de un *stakeholder* implica en realidad, guardar el conjunto de elecciones que ha realizado los *stakeholders* como respuestas. Cada respuesta está asociada a una encuesta determinada.

Precedence Diagram

Se refiere a un objeto de StarUML, específicamente a un diagrama, del que se obtendrá información de los casos de uso y las relaciones entre ellos, como entrada para generar su secuencia de construcción.

Use Case

Se refiere a un grupo de objetos de StarUML de los que está compuesto un diagrama de precedencias.

BuildSecuence

Se refiere al resultado de la aplicación del algoritmo de generación de secuencia de construcción. Se añade al proyecto seleccionado como un objeto de StarUML.

UserType

Se refiere al tipo de usuario que contesta la encuesta. Dado que hemos visto que no todos los usuarios tienen el mismo peso, esta entidad se refiere a los tipos de usuarios identificados, con sus pesos correspondientes. Cada usuario que llene la encuesta deberá especificar a qué tipo de usuario pertenece.

5.1.3. Listado de operaciones

Según la metodología adoptada, el segundo entregable de esta etapa es un listado de las operaciones del módulo. Estas operaciones implican la realización funcional de un caso de uso en cada uno de sus escenarios. En la tabla 2 se muestra esta información:

Operación	Caso de Uso asociado
Añadir pregunta	CU Añadir pregunta
Llenar encuesta	CU Llenar encuesta
Generar secuencia de construcción	CU Generar secuencia de construcción

Tabla 2. Elementos involucrados y sus funciones.

Como puede notarse, se ha identificado una sola operación por cada Caso de Uso del modelo.

5.2. Modelo utilizado para el procesamiento de encuestas

A continuación, se presenta el algoritmo que utilizamos para procesar las encuestas respondidas, y cruzar esta información con la secuencia obtenida por el diagrama de precedencias. Luego de capturar la opinión de los *stakeholders*, mediante las encuestas, se inicia su procesamiento.

Para esto, seguiremos estos pasos:

1. Recuperar las encuestas llenas
2. Promediar a un valor por pregunta los valores obtenidos
3. Promediar a un valor por C.U. los valores obtenidos
4. Establecer orden de C.U. por encuestas
5. Recuperar C.U. de los diagramas de precedencias
6. Establecer orden de C.U. por precedencias
7. Promediar órdenes calculados
8. Establecer orden final de C.U.

La figura 14 presenta el diagrama de actividades para clarificar este proceso:

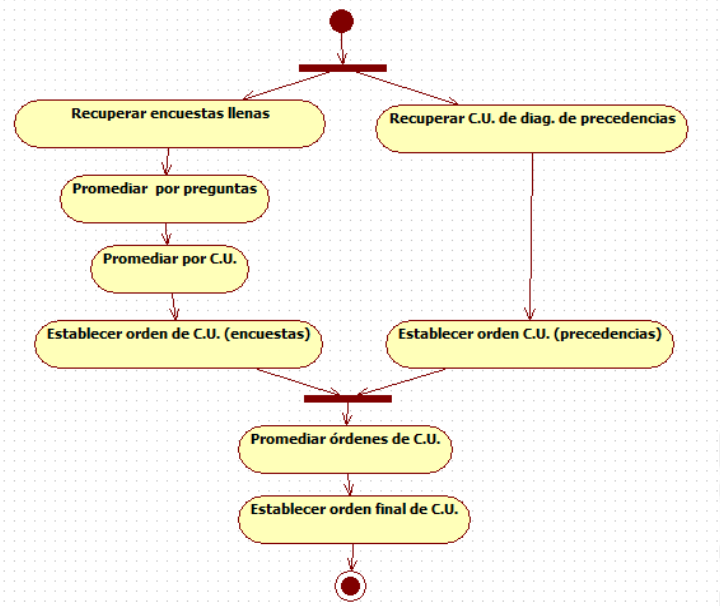

Figura 14. Algoritmo para procesar encuestas (diagrama de actividades)

Para ejemplificar lo anterior, a continuación se enuncia y desarrolla un ejercicio.

> *Se tiene tres CU: CU1, CU2, CU3. El primero y el tercero tienen asociadas dos preguntas, el segundo tiene una sola. Las encuestas están conformadas entonces por cinco preguntas. Cada pregunta tiene un peso diferente.*
>
> *Se ha aplicado la encuesta a cuatro usuarios. Quienes respondieron las encuestas tienen pesos diferentes. Las encuestas llenas se identifican como E1, E2, E3 y E4.*

La figura 15 muestra gráficamente las encuestas llenas.

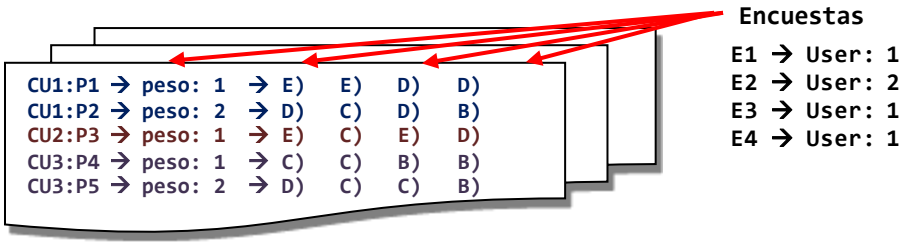

Figura 15. Ejemplo de encuestas por procesar y sus respuestas.

Entonces, se procede a aplicar el modelo:

1. Recuperar las encuestas llenas
   Se traslada a la tabla 3 la información con la que se cuenta de las encuestas.

CASO DE USO	PREGUNTA	PESO PREGUNTA	ENCUESTA / PESO USUARIO			
			E1	E2	E3	E4
			1	2	1	1
CU1	P1	1	-2	-2	-1	-1
CU1	P2	2	-1	0	-1	1
CU2	P3	1	-2	0	-2	-1
CU3	P4	1	0	0	1	1
CU3	P5	2	-1	0	0	1

Tabla 3. Obtener encuestas.

2. Promediar por preguntas
   Se utiliza el promedio ponderado, para obtener un valor único por cada pregunta.
   `(P1*PESO_P1 + P2*PESO_P2+…+Pn*PESO_Pn) / (PESO_P1 + … + PESO_Pn)`

CASO DE USO	PREG	PROMEDIAR PREGUNTAS						
		OPCIONES * PESO USUARIO				SUMA ANTERIOR	SUMA PESO USUARIOS	PROMED PREG
		OPC_E1 * PESO1	OPC_E2 * PESO2	OPC_E3 * PESO3	OPC_E4 * PESO4	SUMA(OPC_En*PESOn)	SUMA(En)	DIVISIÓN
CU1	P1	-2	-4	-1	-1	-8	5	-1.6
CU1	P2	-1	0	-1	1	-1		-0.2
CU2	P3	-2	0	-2	-1	-5		-1
CU3	P4	0	0	1	1	2		0.4
CU3	P5	-1	0	0	1	0		0

Tabla 4. Promediar por preguntas.

3. Promediar por C.U.
   Se utiliza el promedio ponderado para obtener un valor único por CU.
   `(OPC_E1*PESO1 + OPC_E2*PESO2 + OPC_E3*PESO3 + OPC_E4*PESO4) / (PESO1 + PESO2 + PESO3 + PESO4)`

CASO DE USO	PREG	PESO PREG	PROMEDIO PREGUNTA	PROMEDIAR CASOS DE USO			
				PESO_Pn * Pn	SUMA (PESO_Pn * Pn)	SUMA PESO_Pn	PROMED CU
							DIVISIÓN
CU1	P1	1	-1.6	-1.6	-2	3	-0.667
	P2	2	-0.2	-0.4			
CU2	P3	1	-1	-1	-1	1	-1
CU3	P4	1	0.4	0.4	0.4	3	0.133
	P5	2	0	0			

Tabla 5. Promediar por C.U.

4. Establecer orden de C.U. por encuestas
   De los cálculos anteriores, se obtiene el orden en la tabla 6.

CASO DE USO	PROMEDIO CU
CU3	0.133
CU1	-0.667
CU2	-1

Tabla 6. Orden (por encuesta)

5. Recuperar C.U. de los diagramas de precedencias, como muestra la figura 16.

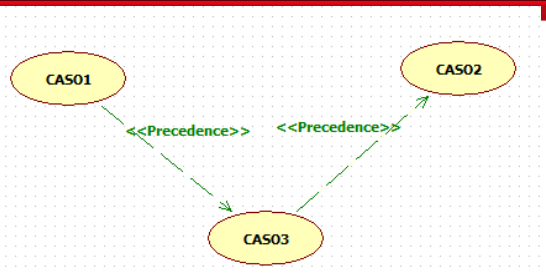

Figura 16. Diagrama de precedencias para el ejemplo propuesto.

6. Establecer orden de C.U. por precedencias
   De la figura 15, se da pesos límite -2, 0 y +2 a los C.U. recuperados[6]:

CASO DE USO	PROMEDIO PRECED
CU1	2
CU3	0
CU2	-2

Tabla 7. Orden (por encuesta)

7. Promediar órdenes calculados
   Se promedian los dos órdenes calculados. Para este modelo, se considera que el orden por precedencias tiene 50% de más peso, por tanto:
   `(PROM_CU + 1.5*PROM_PRE) / 2.5`

CASO DE USO	PROMEDIO CU	PROMEDIO PRECED	RESULTADO
CU1	-0.667	2	0.933
CU2	-1	-2	-1.6
CU3	0.133	0	0.053

Tabla 8. Orden (por encuesta)

8. Establecer orden final de C.U.
   Por último, se ordena de mayor a menor, y se tiene:

CASO DE USO	PROMEDIO FINAL
CU1	0.933
CU3	0.053
CU2	-1.6

Tabla 9. Orden (por encuesta)

5.2.1. Generalización del modelo

Del algoritmo anterior presentado, se generaliza el modelo:

Entidades:		Pesos:		Cantidades:	
Pregunta:	Q	Peso de la pregunta:	p	Cantidad de encuestas llenas:	n
Opción:	O	Peso de la opción:	q	Cantidad de preguntas asoc. por CU:	m
CU:	K	Peso del stakeholder:	s	Cantidad de CU:	c
		Peso del CU por precedencia:	R		
		Peso del CU por encuesta:	C		
		Peso del CU final:	F		

[6] *Estos valores se explican en la sección 5.2.2*

Se obtiene las siguientes generalizaciones:

$$Q = \frac{\sum_{1}^{n} (qi)(si)}{\sum_{1}^{n} (si)}$$

$$C = \frac{\sum_{1}^{m} (Qi)(pi)}{\sum_{1}^{m} (pi)}$$

$$F = \frac{(C) + (1.5)(R)}{2.5}$$

5.2.2. Criterios adicionales del modelo

- Todas las opciones el valor predeterminado de +2 +1 0 -1 -2
- Los pesos de los usuarios serán desde 1, 2, en adelante; en función del poder de decisión sobre el *software*. Para este caso ejemplo, se tiene al usuario *Executive* con peso 1, y al usuario *Management* con peso 2.
- Los pesos de las preguntas son 1, 2, 3 según sean easy, medium y hard
- Al recuperar el orden de los CU en las precedencias, el criterio será el siguiente: Poner al primer y al último CU, el valor de +2 y -2, e ir completando los valores intermedios, basándose en la fórmula del n-ésimo término de una sucesión aritmética:

  Que, desarrollándola para los primeros casos, se tiene:

$$a_n = a_1 + (n-1)(d)$$

Cantidad de CU	Elementos							an - a1	n - 1	d
1 CU		**0**						4	0	
2 CU	**2**	**-2**						4	1	4
3 CU	**2**	0	**-2**					4	2	2
4 CU	**2**	0.67	-0.67	**-2**				4	3	1.33
5 CU	**2**	1	0	-1	**-2**			4	4	1
6 CU	**2**	1.2	0.4	-0.4	-1.2	**-2**		4	5	0.8
7 CU	**2**	1.33	0.67	0	-0.67	-1.33	**-2**	4	6	0.67

Tabla 10. Pesos de CU desde sus precedencias.

Se puede observar este mismo ejemplo, desarrollado dentro de la herramienta terminada, en el **Capítulo 7: Pruebas**.

Capítulo 6: Arquitectura y Diseño

En el presente capítulo, se desarrolla la arquitectura con la que se ha trabajado el módulo. Además, se presenta el diseño utilizado para el mismo.

Como ya se mencionó, se ha trabajado extendiendo el proyecto original del TUPUX.

6.1. Arquitectura del módulo

El módulo implementado tiene la arquitectura mostrada en la figura 17.

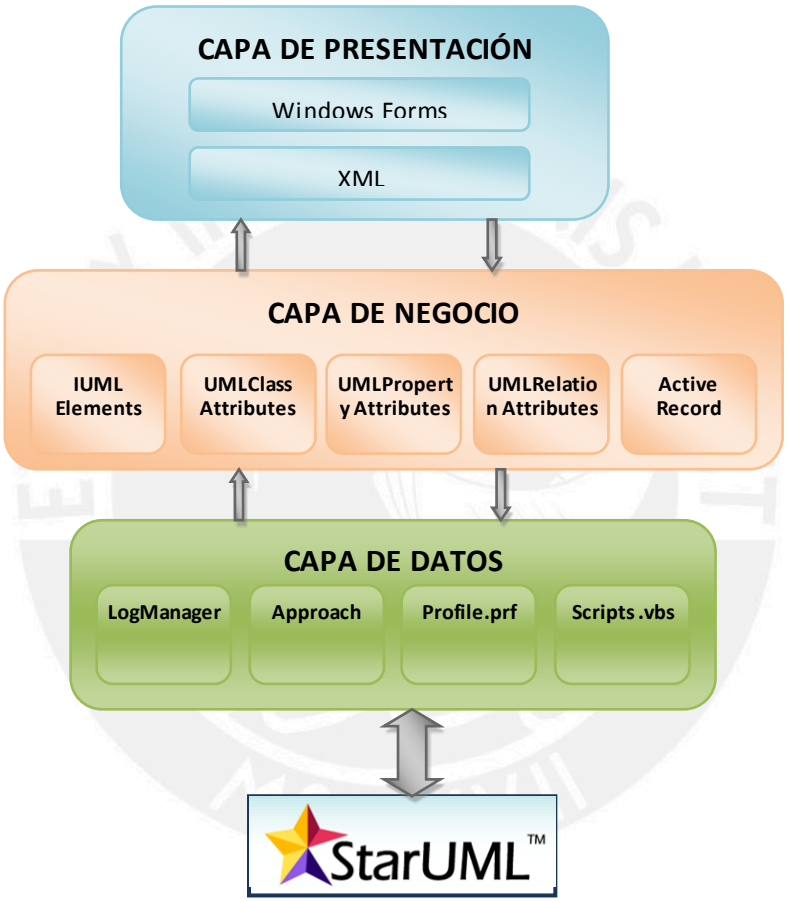

Figura 17. Arquitectura del módulo.

Como se puede apreciar, la arquitectura del módulo tiene tres capas:

La capa de presentación, formada por los medios por los cuales el usuario puede interactuar con la aplicación. En este caso, los componentes IU, y los archivos XML.

La capa de negocio, la cual encapsula la lógica, los algoritmos, y en general, la forma en la que se manipula y procesa la información intercambiada entre el usuario y el StarUML.

La capa de acceso a datos, la que contiene clases que interactúan con el origen de datos. Esta parte es especial, ya que no hay conexión con una base de de datos, sino que se interactúa con clases de la API del StarUML, que permiten manipular cualquier clase dentro del proyecto activo.

6.1.1. Proyecto en Visual Studio 2008

El módulo ha sido desarrollado sobre el proyecto TUPUX, pues utiliza intensivamente clases ya preparadas por este.

En ese sentido, se puede ver cómo el TUPUX ha crecido en sus diferentes clases (separadas en sus respectivos proyectos) en la figura 18.

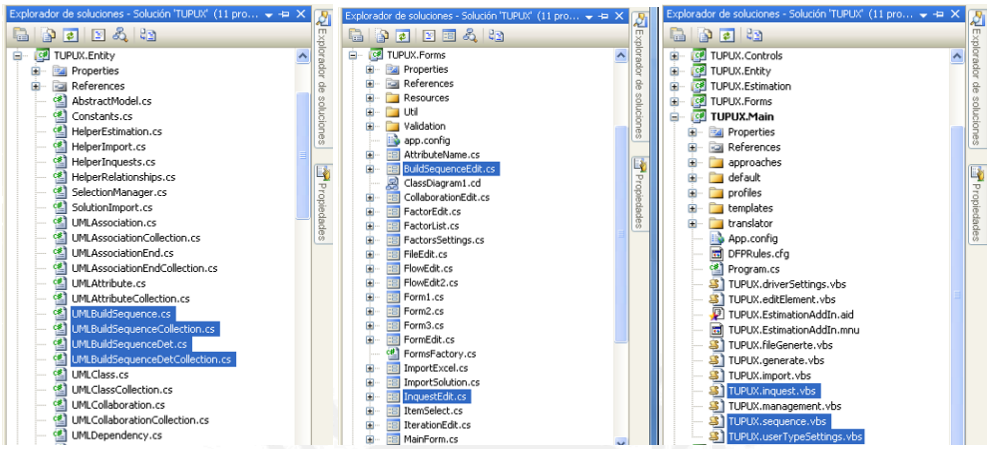

Figura 18. Solución TUPUX y nuevos componentes (Captura de pantalla VS2008)

Así mismo, sobre esta base se creó el ejemplo para cargar aplicaciones por fuera. De esta manera se ha ejemplificado cómo los archivos XML permiten intercambiar información con otras aplicaciones externas.

Las principales clases implementadas y sus principales atributos pueden consultarse en los anexos del presente documento.

6.1.2. Formato de los archivos XML de encuestas

Para intercambiar información, se usan archivos XML *standalone*[7]. Esto hace referencia a que el archivo XML se lee solo, sin dependencia de otros archivos para su procesamiento.

El formato de la plantilla (encuesta vacía) se muestra en la figura 19:

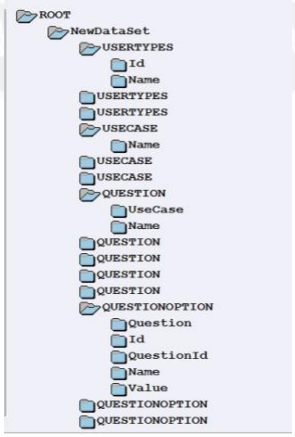

Figura 19. Árbol del XML de encuesta vacía[8]

Se puede observar que se ha definido los tipos de usuario, los casos de uso, las preguntas y las opciones, cada uno con sus atributos.

[7] *http://www.w3.org/XML/*
[8] *Generado desde http://www.jeremie.com/Dev/XML/test/index.html*

Así mismo, el formato de las encuestas llenas es el mostrado en la figura 20:

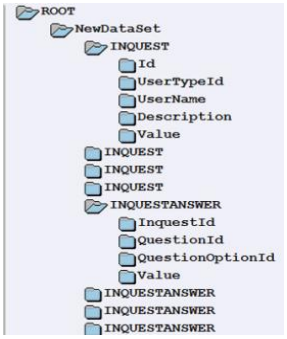

Figura 20. Árbol del XML de encuestas llenas

Se puede observar que se ha guardado las encuestas y sus respectivas respuestas.

6.1.3. Despliegue

Una vez compilado el proyecto, este debe copiarse dentro de la ruta de instalación del *StarUML*, carpeta /modules/ como muestra la figura 21.

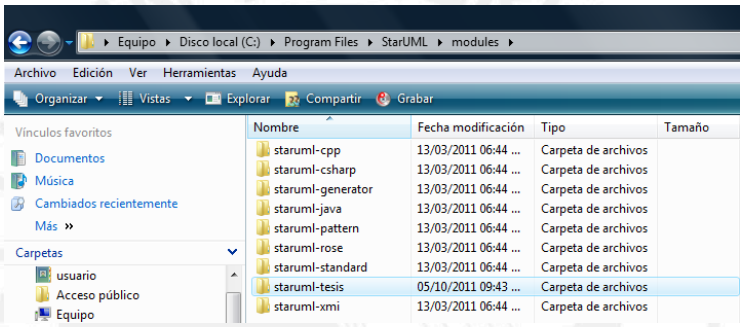

Figura 21. Ubicación del módulo compilado (captura de pantalla)

Con esto se consigue, que al lanzar el *StarUML*, se cargue el módulo. (Cabe recordar que el módulo ha sido desarrollado sobre el TUPUX). La figura 22 muestra el módulo cargado al abrir el *StarUML*.

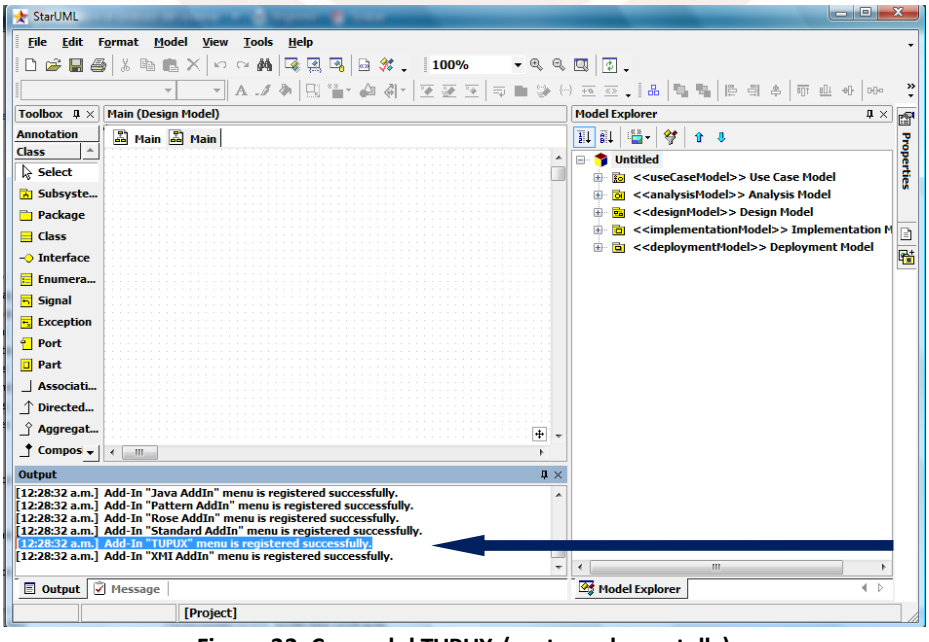

Figura 22. Carga del TUPUX (captura de pantalla)

6.2. Diseño

Para proseguir con el proceso de construcción de *software*, se desarrollamos la fase de Diseño según la metodología adoptada. Para esta fase, se debe documentar información relativa a:

- Definición de las operaciones identificadas
- Desarrollo de una función que cubra la operación identificada
- Prototipos de las interfaces de usuario requeridas

6.2.1. Operaciones

Añadir pregunta
Que viene del caso de uso **Añadir pregunta.**
Se le asocia una función con el mismo nombre.

Llenar encuesta
Que viene del caso de uso **Llenar encuesta.**
Se le asocia una función con el mismo nombre.

Generar secuencia de construcción
Que viene del caso de uso **Generar secuencia de construcción.**
Se le asocia una función con el mismo nombre.

Para cada función, se modela su funcionalidad en *StarUML*, y luego se documenta aquí:

- El diagrama de Secuencias
- El diagrama de clases de diseño
- Las interfaces GUI prototipo

6.2.2. Operación: Añadir Pregunta

Mediante esta operación, se lleva a cabo el C.U. del mismo nombre. Aquí se tiene la gestión de todo lo concerniente a la creación de las encuestas. En las figuras 23 y 24 se muestran los diagramas de secuencias y de clases de diseño, respectivamente.

Diagrama de secuencias

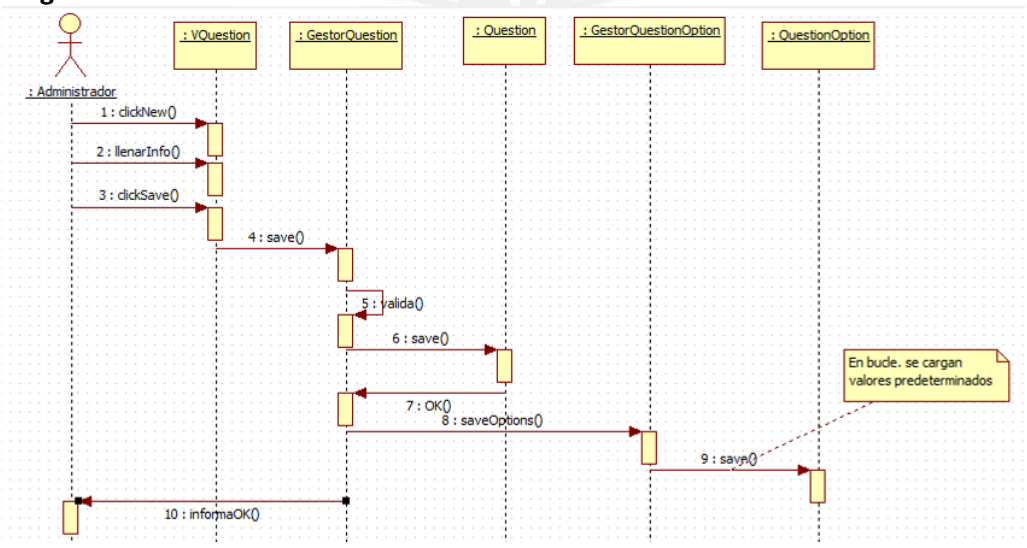

Figura 23. Diagrama de secuencias: Añadir pregunta

Diagrama de clases de diseño

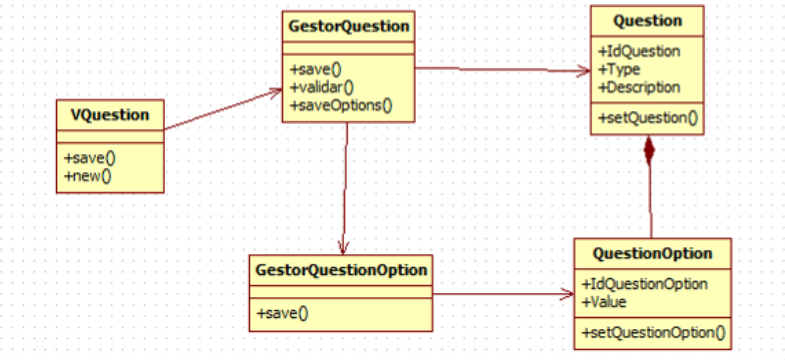

Figura 24. Diagrama de clases de diseño: Añadir pregunta

Interfaz GUI

La forma de asociar preguntas a CU es ingresar por la opción TUPUX Edit, e ir a la última pestaña: Questions. En esta pestaña se podrá dar mantenimiento a las preguntas asociadas al CU seleccionado. La figura 25 muestra la interfaz GUI asociada.

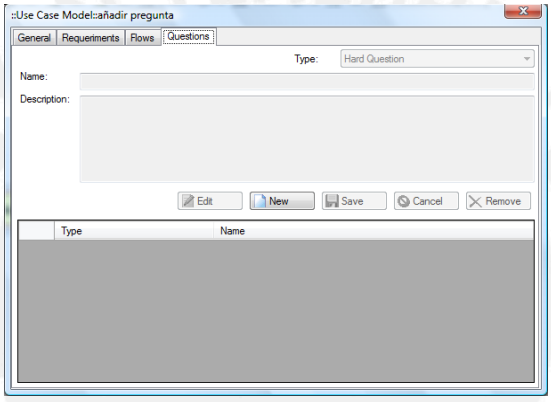

Figura 25. Interfaz GUI: Añadir pregunta (Captura de pantalla)

Al dar clic sobre el botón New, los campos se desbloquean. Se agrega la información requerida. Luego de procede al guardado vía el botón Save. En ese momento, se añaden la pregunta y sus cinco opciones al Project actual. La figura 26 muestra lo especificado.

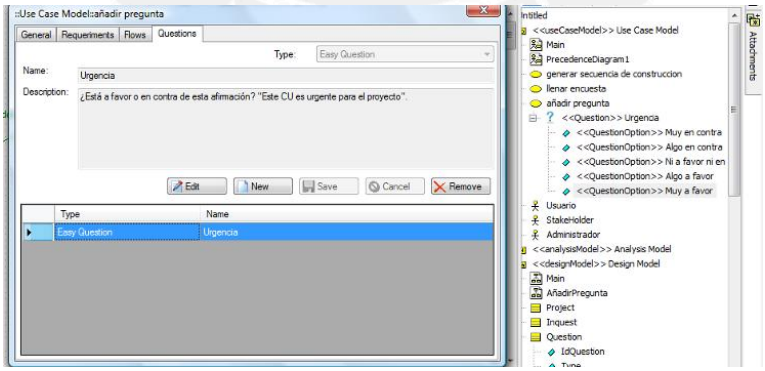

Figura 26. Utilización de GUI añadir pregunta en StarUML (Captura de pantalla)

En la figura 27, aparecen todas las preguntas que se creen para el Caso de Uso seleccionado. Las opciones, así mismo, se pueden personalizar en sus textos y en sus valores.

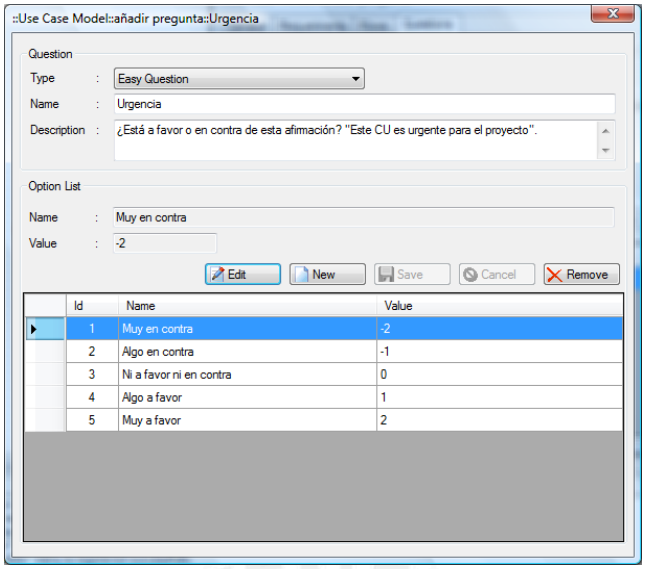

Figura 27. GUI para editar preguntas y sus valores en StarUML (Captura de pantalla)

6.2.3. Operación: Llenar encuesta

Mediante esta operación, se lleva a cabo el C.U. del mismo nombre. Aquí se tiene la gestión de todo lo concerniente al llenado de las encuestas por parte de los *stakeholders*. Las figuras 28 y 29 muestran los diagramas de secuencias y de clases de diseño respectivamente.

Diagrama de secuencias

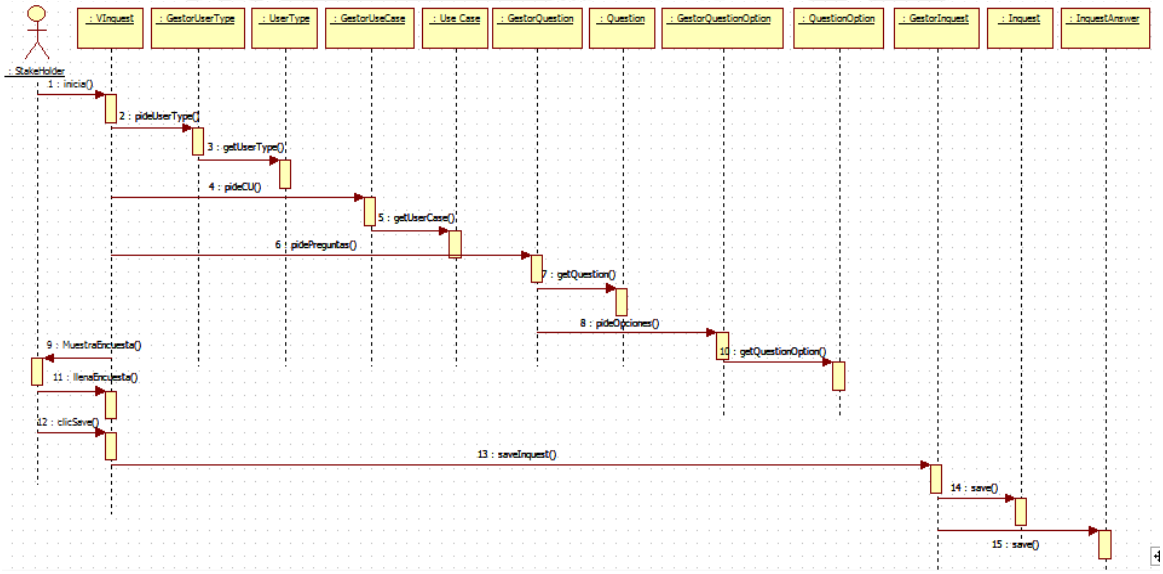

Figura 28. Diagrama de secuencias: Llenar encuesta

Diagrama de clases de diseño

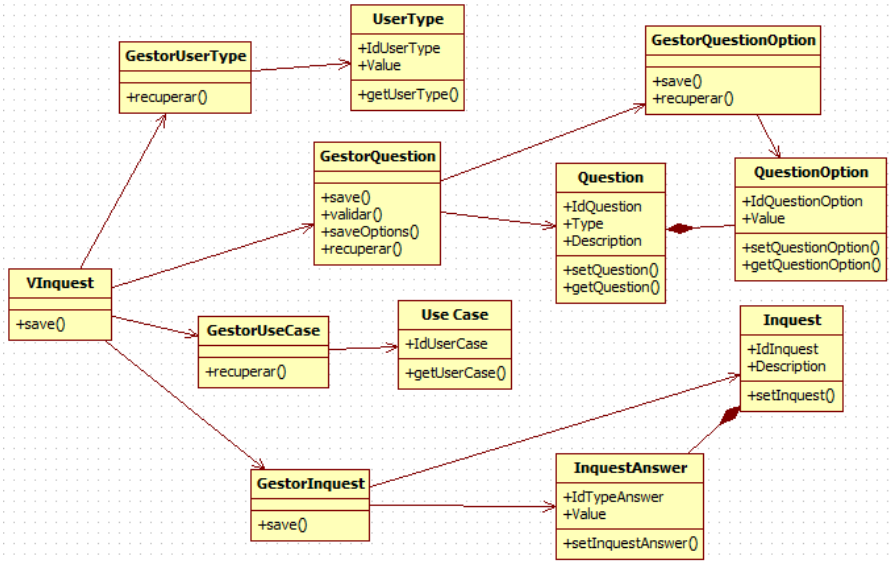

Figura 29. Diagrama de clases de diseño: Llenar encuesta

Interfaz GUI

Para llenar una encuesta, se debe seleccionar la opción del menú correspondiente para que se proceda a generar la encuesta.

Al dar clic, el módulo recupera todas las preguntas de todos los CU y genera una interfaz para que el usuario pueda llenar su encuesta, tal como se muestra en la figura 30.

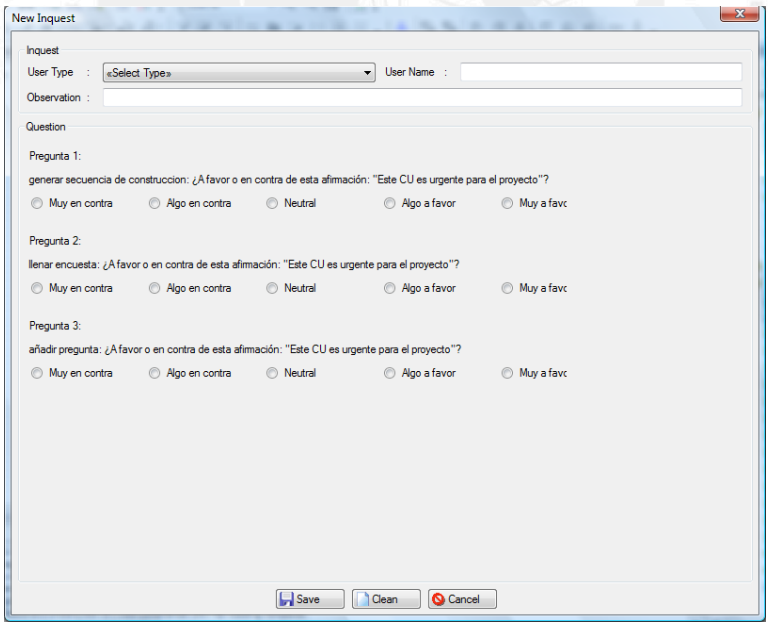

Figura 30. GUI Llenar encuesta (Captura de pantalla)

Una vez el usuario ha llenado la información solicitada, da clic en el botón Save, y se procede a guardar la encuesta dentro del proyecto. Al llenar la encuesta, el usuario ha podido colocar su nombre, una observación, y sobre todo, ha colocado el tipo de usuario al cual él pertenece; el cual será un dato importante en el procesamiento de las encuestas. La figura 31 muestra cómo se guarda la información de la encuesta llena dentro del *StarUML*.

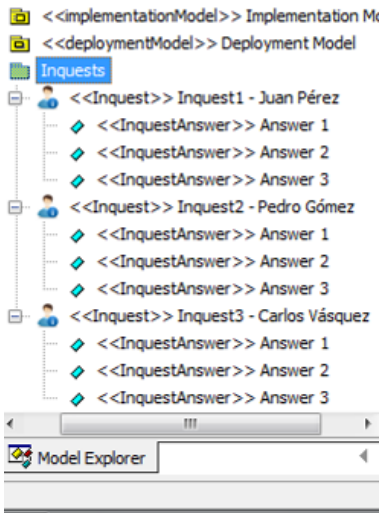

Figura 31. Encuesta llena guardada dentro el StarUML (Captura de pantalla)

Cabe destacar que también es posible llenar las encuestas por fuera de la herramienta. Para esto, están los botones Importa Data y Export Data mostrados en la figura 32. El primero genera la encuesta hacia un archivo XML que puede ser cargado en alguna herramienta externa. El segundo, carga la encuesta llena, vía el mismo procedimiento. Para este caso, se tiene una herramienta ejemplo que carga el archivo XML generado por el *StarUML*, permite grabar la encuesta como si se estuviera en el *StarUML*, y genera otro archivo, que posteriormente se carga al *StarUML*.

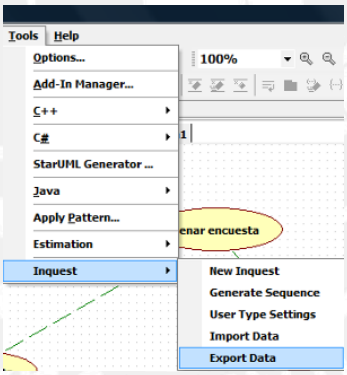

Figura 32. Botones de exportación e importación de encuestas a archivo XML (Captura de pantalla)

La figura 33 muestra cómo mediante una aplicación de ejemplo, se puede llenar las encuestas por fuera del *StarUML*.

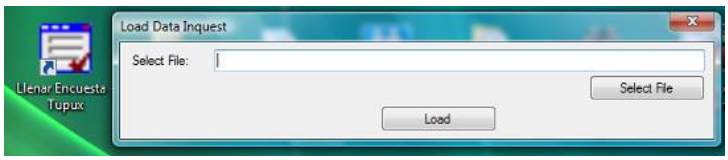

Figura 33. Aplicación de ejemplo para llenar encuestas por fuera (Captura de pantalla)

En la figura 34 se observa el llenado de una encuesta desde fuera del *StarUML*. Nótese que la interfaz es la misma, puesto que se usan las mismas ventanas base del .NET *Framework*.

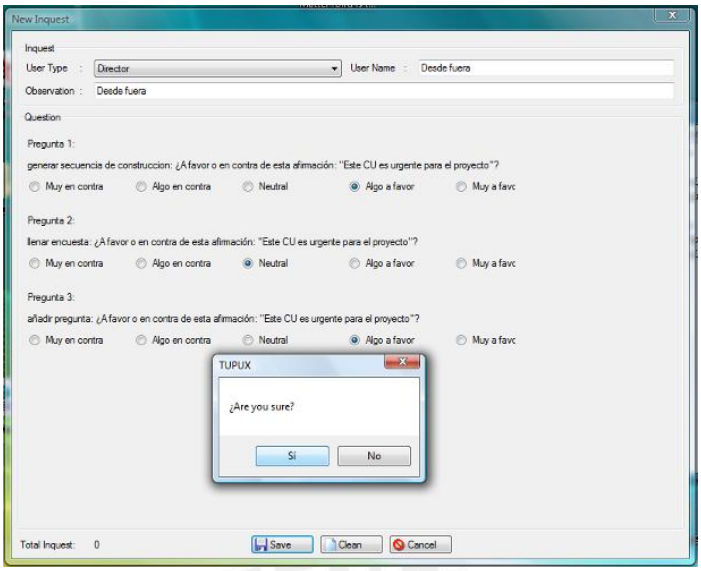

Figura 34. Llenado de encuesta por fuera (Captura de pantalla)

La figura 35 muestra los archivos XML de intercambio generados.

Figura 35. Archivo con encuestas llenas generado (Captura de pantalla)

En la figura 36 vemos la manera de importar las encuestas ya llenadas por fuera de la herramienta.

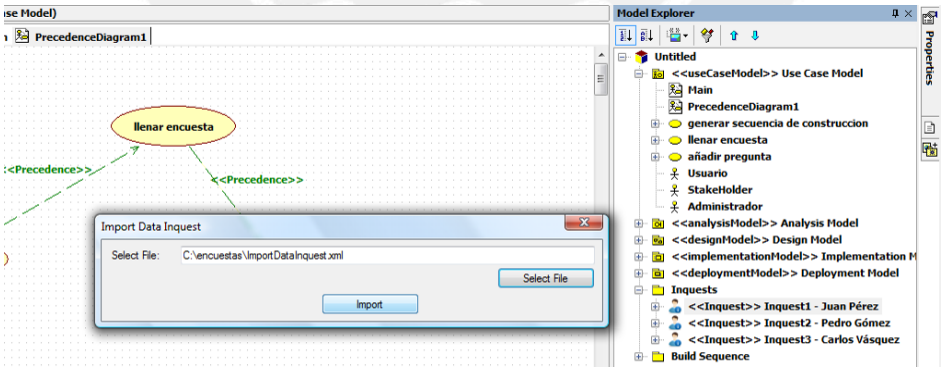

Figura 36. Carga de las encuestas llenas vía *StarUML* (Captura de pantalla)

6.2.4. Operación: Generar secuencia de construcción

Mediante esta operación, se lleva a cabo el C.U. del mismo nombre. Aquí se tiene el procesamiento de las encuestas y su cruce con los valores recuperados de los diagramas de precedencia. Las figuras 37 y 38 muestran los diagramas de secuencias y de clases de diseño respectivamente.

Diagrama de secuencias

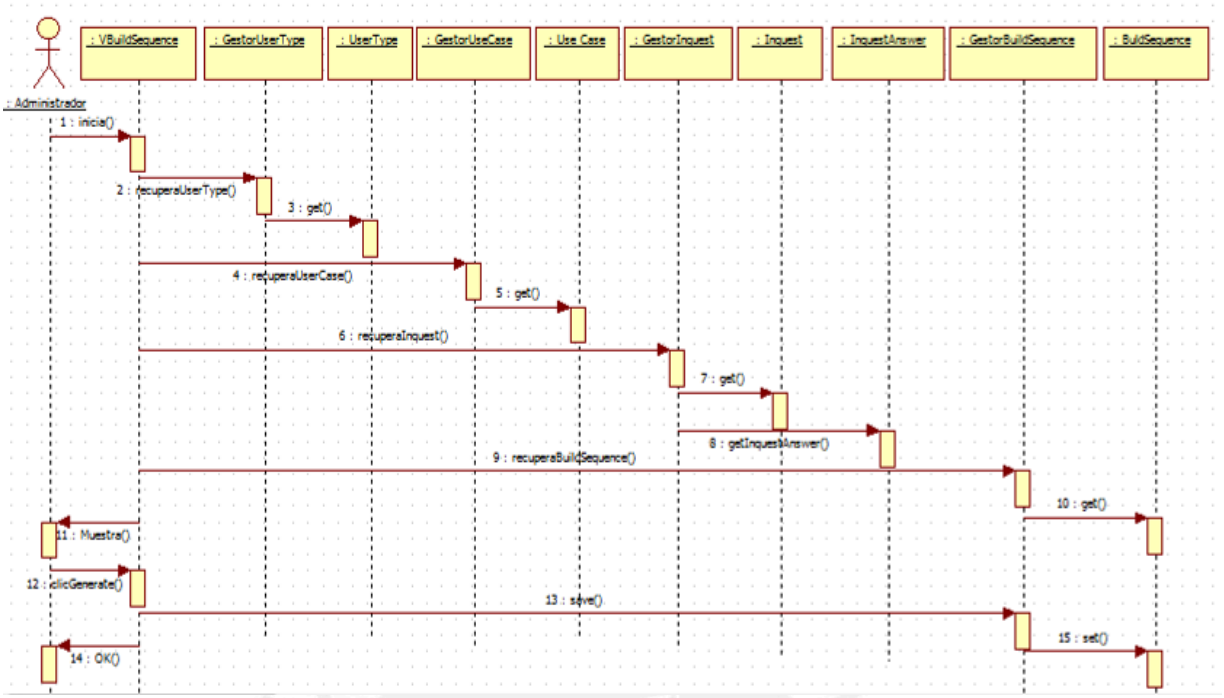

Figura 37. Diagrama de secuencias: Generar secuencia de construcción

Diagrama de clases de diseño

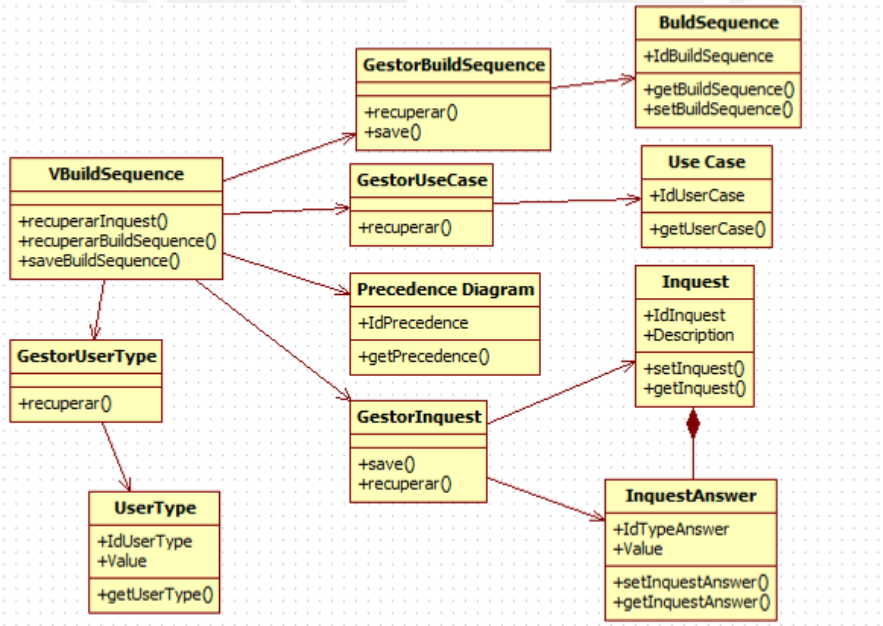

Figura 38. Diagrama de clases de diseño: Generar secuencia de construcción

Interfaz GUI

Para generar la secuencia de construcción, se debe seleccionar la opción del menú correspondiente para que se proceda con dicha tarea.

Al dar clic en el botón para generar la secuencia, se muestra una pantalla donde se han recuperado todas las encuestas llenas, y en caso ya haya existido una secuencia previa, se carga también. La figura 39 muestra esta operación.

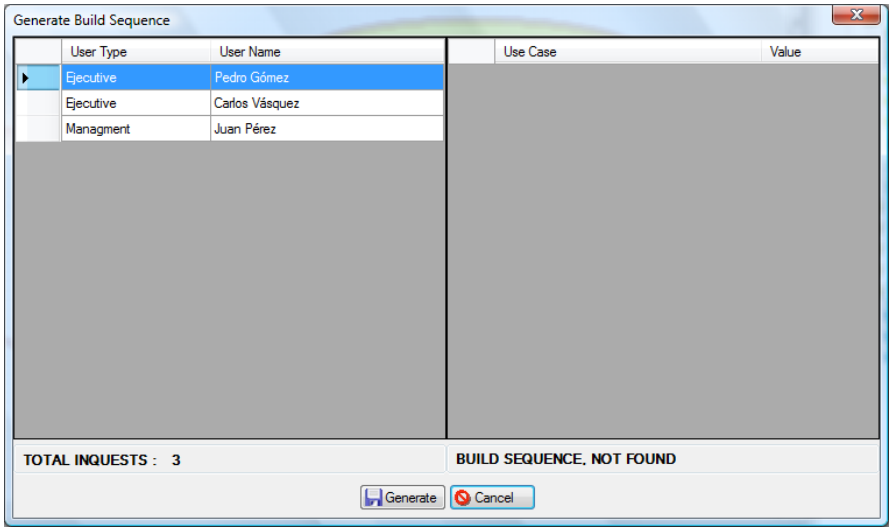

Figura 39. GUI generar secuencia de construcción (Captura de pantalla)

Se da clic en el botón Generate, y se puede observar cómo en la parte izquierda se ha cargado el resultado del procesamiento descrito en la sección correspondiente. La figura 40 muestra la GUI cargada y los datos guardados dentro del *StarUML*. Se puede notar además, que en el explorador del modelo, se ha agregado un Package adicional, con la Build Sequence generada.

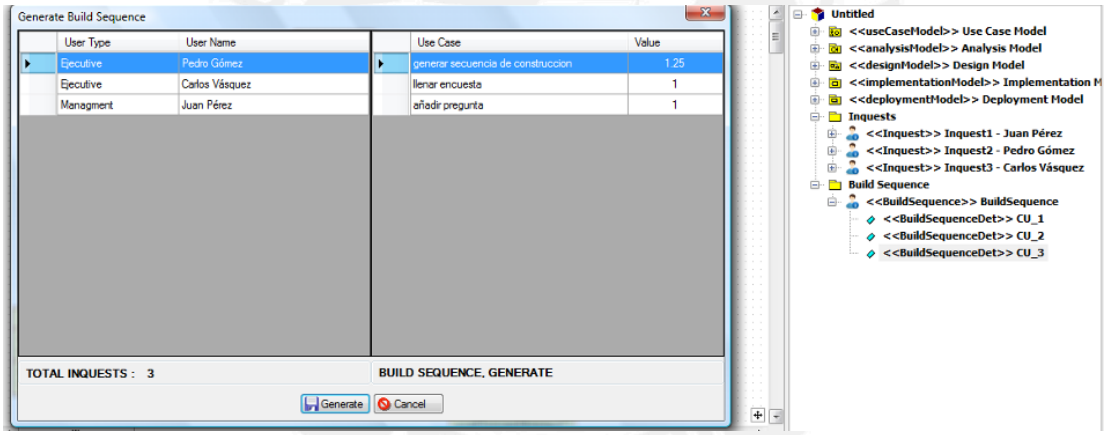

Figura 40. Secuencia generada y datos guardados en *StarUML* (Captura de pantalla)

Capítulo 7: Pruebas

En este capítulo, se pone a prueba el módulo desarrollado. Para esto, se trae el ejemplo planteado para definir el modelo de procesamiento de encuestas. La prueba del módulo, es que debe arrojar la misma información que se ha calculado manualmente.

Así mismo, se recorre cada C.U. para identificar la funcionalidad, y probar si es que el módulo trabaja según lo esperado.

En ese sentido, se vuelve al enunciado de la sección 5.2:

> *Se tiene tres CU: CU1, CU2, CU3. El primero y el tercero tienen asociadas dos preguntas, el segundo tiene una sola. Las encuestas están conformadas entonces por cinco preguntas. Cada pregunta tiene un peso diferente.*
>
> *Se ha aplicado la encuesta a cuatro usuarios. Quienes respondieron las encuestas tienen pesos diferentes. Las encuestas llenas se identifican como E1, E2, E3 y E4.*

En ese sentido a la figura 15 se añaden los pesos de cada pregunta, como muestra la figura 41.

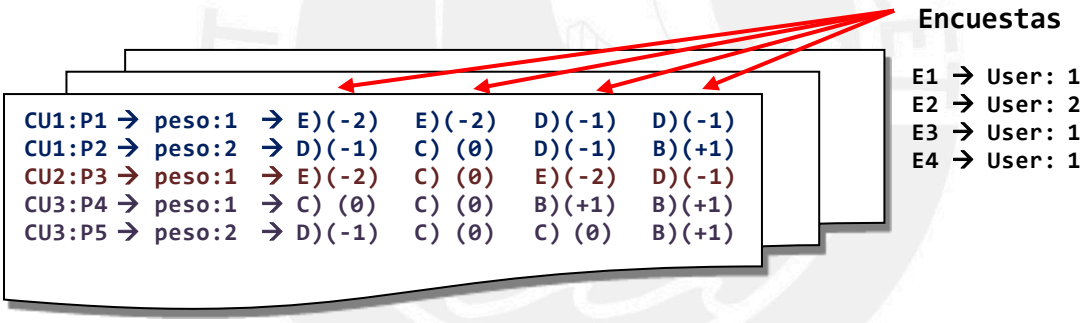

Figura 41. Encuestas llenas con sus respectivos pesos.

Entonces, se procede a desarrollar este ejemplo propuesto dentro de la herramienta, según la secuencia lógica de uso:

1. Elaboración de la encuesta (C.U. Añadir pregunta)
2. Llenado de la encuesta (C.U. Llenar encuesta)
3. Procesamiento de las encuestas (C.U. Generar secuencia de construcción)

Caso de Uso: Añadir pregunta

Dentro de la herramienta, se ha modelado los tres CU, con sendas preguntas, como se muestra en la figura 42.

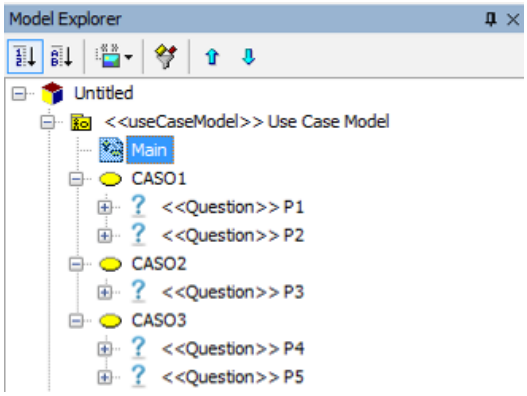

Figura 42. Encuesta y preguntas dentro del *StarUML* (Captura de pantalla)

Ahora, se verifican los tipos de usuario configurados, como muestra la figura 43:

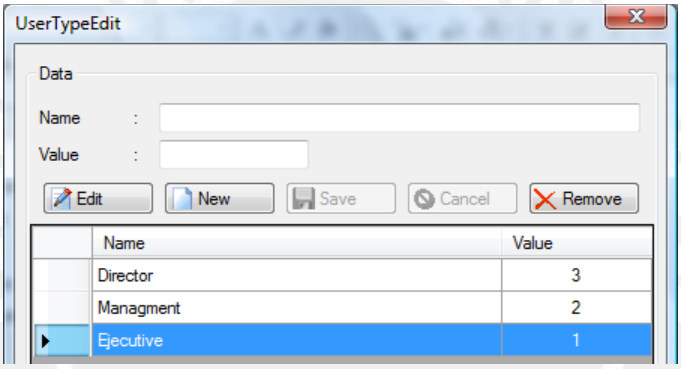

Figura 43. Usuarios configurados con sus pesos dentro del *StarUML* (Captura de pantalla)

Caso de uso: Llenar encuesta

La figura 44 muestra la exportación de la encuesta, para ser llenada desde fuera del *StarUML.*:

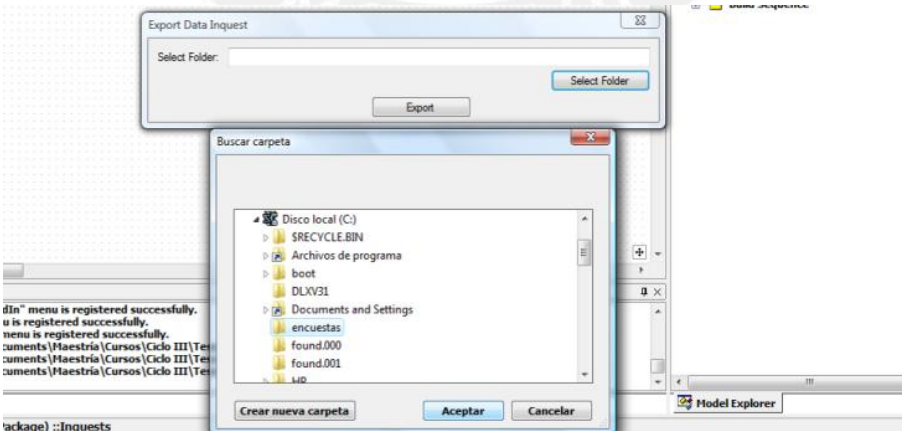

Figura 44. Exportación de la encuesta desde el *StarUML* (Captura de pantalla)

Se llena las cuatro encuestas, según los valores dados, según se muestra en la figura 45.

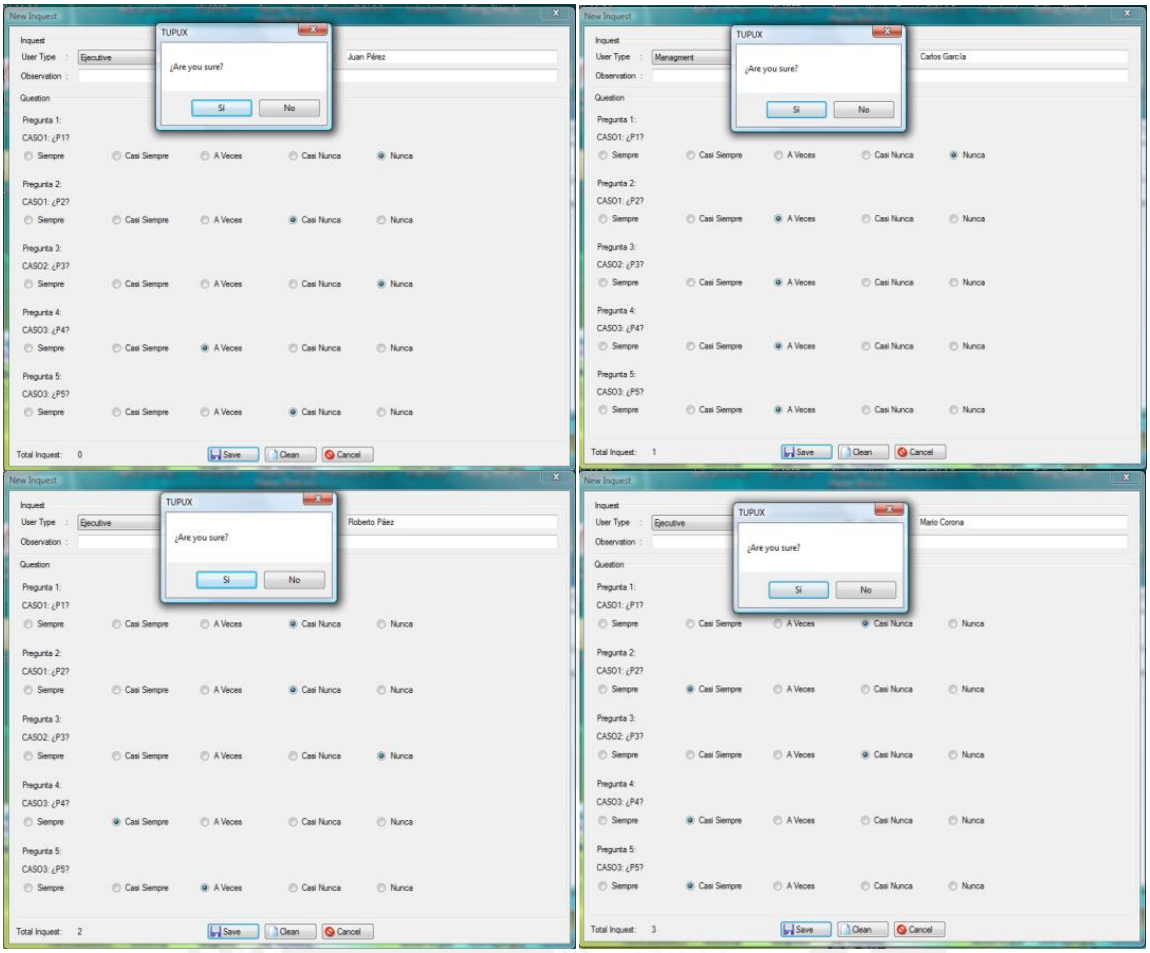

Figura 45. Encuestas llenas desde fuera del *StarUML* (Captura de pantalla)

En la figura 46 se muestra el proceso de cargar las encuestas llenas (y guardadas en archivos XML) al *StarUML*.

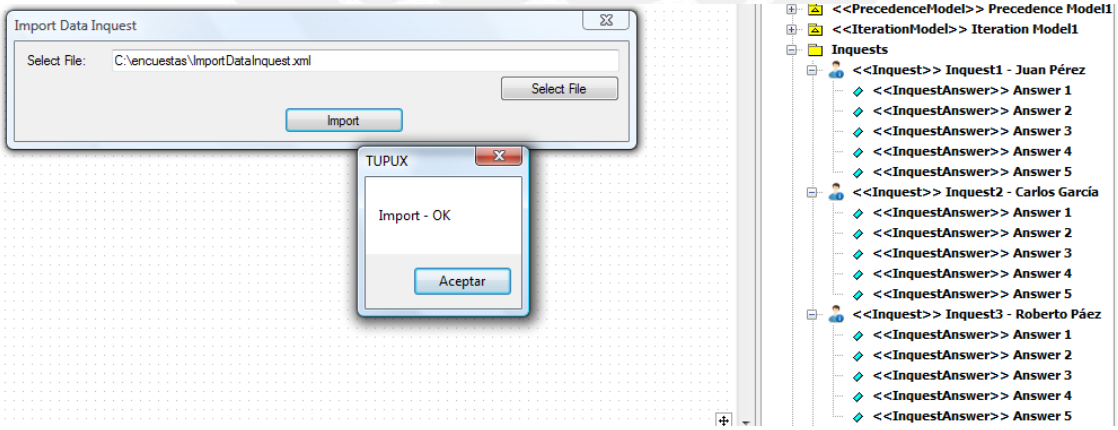

Figura 46. Carga de encuestas llenas al *StarUML* (Captura de pantalla)

Caso de uso: Generar secuencia de construcción

La figura 47 muestra la operación de generar la secuencia de construcción. Como se puede observar, los valores arrojados por el módulo, son los mismos que los calculados manualmente.

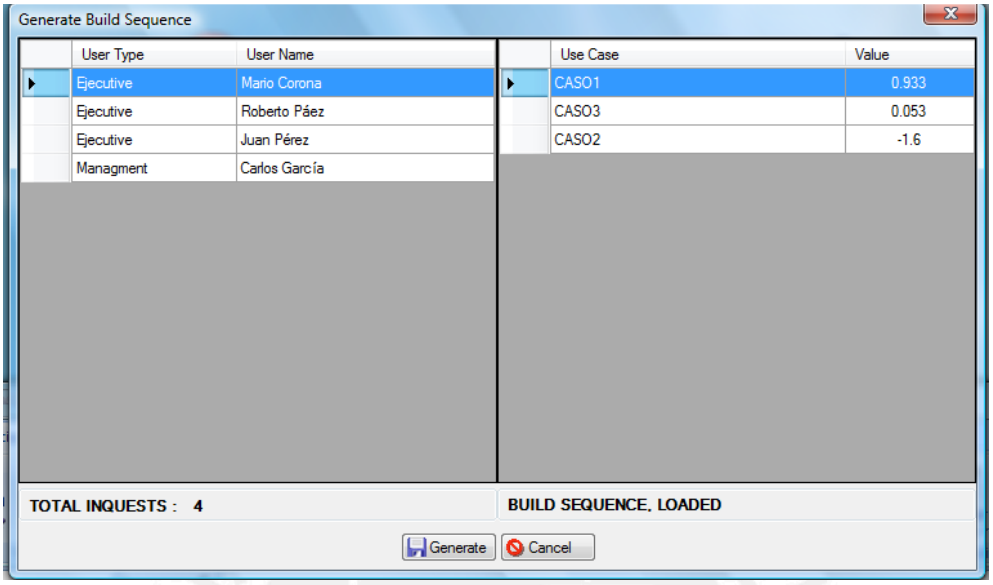

Figura 47. Secuencia de Construcción (Captura de pantalla)

Así mismo, se ha tenido la oportunidad de probar funcionalidad sobre los tres casos de uso presentados, al seguir la secuencia lógica de utilización de la herramienta.

En ese sentido, se puede afirmar que no solo que se están cumpliendo los requerimientos, sino que el modelo implementado está en funcionamiento y arroja resultados consistentes con los esperados.

Capítulo 8: Conclusiones y recomendaciones

Luego del desarrollo del presente trabajo, se puede esbozar algunas conclusiones:

- El módulo desarrollado permite recoger y tomar en cuenta la voz de los stakeholders, en el momento de definir la secuencia de construcción de casos de uso; ya que estas opiniones, necesidades, expectativas y experiencias constituyen verdaderas restricciones con respecto al sistema que se desarrolla, y en ese sentido, impactan tarde o temprano en el proceso de construcción del software que se está desarrollando. En ese sentido, el presente módulo permite capturar, sistematizar y utilizar dicha información en beneficio del proyecto de software dado.

- El módulo desarrollado como un plug-in de la herramienta StarUML, mediante la explotación de la API que la misma pone a disposición, muestra cómo esta herramienta no solo se puede usar para el modelado, sino que puede ser aprovechada para capturar información adicional que enriquezca el proceso mismo de desarrollo de software.

- El uso de una metodología simplificada con casos de uso para construir el presente módulo permitió implementar las funcionalidades propuestas, de manera ordenada, eficiente y documentada, lo que mejora tanto el mantenimiento, la utilización del mismo.

- El ejercicio de automatizar propuestas y nuevas formas de hacer las cosas constituye una herramienta fundamental en la investigación y el desarrollo de la informática, en el sentido de que brinda a quien lo hace no solo los conocimientos de estas nuevas propuestas, sino que lo acerca a uno de los fines de la tecnología, que es llevar a la sociedad, los conocimientos científicos[9].

Así mismo, se propone las siguientes recomendaciones:

- El algoritmo utilizado se basa en general, en promedios lineales de las encuestas capturadas y los pesos de los casos de uso de los diagramas de precedencia. Nuevos algoritmos pueden mejorar el cálculo de los pesos finales, y por tanto, capturar con mayor fidelidad la opinión de los *stakeholders*.

- El presente módulo debe ser probado en proyecto reales, y sus resultados comparados con otros métodos, con el fin de validar sus resultados.

[9] *Núñez, J. La ciencia y la tecnología como procesos sociales. Lo que la educación científica no debería olvidar. OEI en http://www.oei.es/salactsi/nunez02.htm*

- La aproximación de capturar la opinión de los stakeholders por medio de encuestas tiene como punto crítico la generación y selección de las preguntas que conforman las mismas. En ese sentido, es necesario estudiar métodos que permitan elegir las preguntas más adecuadas, así como su redacción y la forma en la que se toman sobre los stakeholders.

- El uso de archivos XML para llenar encuestas por fuera del StarUML puede ser ampliamente explotado; se puede hacer diversas aplicaciones que consuman dichos archivos y que permitan llegar al stakeholder de una manera más oportuna y sencilla para él.

- El presente módulo puede extenderse, para hacerlo más flexible, y sus parámetros más ajustables, sin necesidad de cambios sobre el software. Los mejores valores para estos parámetros deben ser tomados de casos reales.

Anexos

A.1. Catálogo de requisitos

A continuación se muestra en una tabla el catálogo completo de requisitos capturados, que debe cumplir el módulo.

No	Descripción	Tipo	Prioridad
R001	El módulo permitirá generar la secuencia de construcción de casos de uso de acuerdo a los votos de los usuarios.	Funcional	1
R002	El módulo permitirá generar la secuencia de construcción de casos de uso mediante un adecuado algoritmo tomando como base el diagrama de precedencias.	Funcional	1
R003	El módulo permitirá administrar las preguntas de las que se componen las encuestas.	Funcional	2
R004	El módulo permitirá registrar los votos de los usuarios en las encuestas.	Funcional	2
R005	El módulo permitirá generar la ponderación de los votos de los usuarios.	Funcional	1
R006	Se deberá almacenar la secuencia de construcción generada como parte del proyecto en la herramienta	Funcional	2
R007	La herramienta de desarrollo será Visual Studio 2008.	No funcional	1
R008	El sistema estará orientado a ventanas y con diversas opciones de selección	No funcional	1
R009	El lenguaje utilizado uno orientado a objetos y con soporte COM (será C#)	No funcional	1
R010	El módulo deberá ser ejecutado sobre la herramienta StarUML.	No funcional	1
R011	El módulo será manejado bajo mouse y teclado.	No funcional	1
R012	Se debe permitir cargar encuestas llenadas de forma externa al módulo. Esta carga debe seguir alguna forma estándar tipo XML.	No funcional	1

Prioridades:

Número	Descripción
1	Alta
2	Media
3	Baja

A.2. Clases de análisis y sus principales atributos

A continuación se muestra en una tabla, las clases de análisis expuestas en el capítulo 5, con sus principales atributos:

Clases	Atributos	Descripción
Project	IdProject	Número identificador del proyecto
	Name	Nombre del proyecto
	Description	Descripción del proyecto
Question	IdQuestion	Identificador de pregunta
	Type	Tipo de pregunta
	Description	Texto descriptivo de la pregunta
QuestionOption	IdQuestionOption	Texto de opción 1
	IdQuestion	Id de la pregunta a la que está asociada esta entidad
	Value	Valor (peso) parametrizable
Inquest	IdInquest	Identificador de encuesta
	UserTypeId	Tipo de usuario que llenó la encuesta
	UserName	Nombre de usuario que llenó la encuesta
	Description	Observación sobre la encuesta llena
InquestAnswer	IdInquestAnswer	Identificador de respuesta
	IdInquest	Id de la encuestas a la que está asociada esta respuesta
	Value	Opción elegida
UserType	IdUserType	Id del tipo de usuario
	Name	Nombre descriptivo
	Value	Valor (peso) para el tipo de usuario dado
Precedence Diagram	IdDiagram	Identificador del diagrama de precedencia
	IdProject	Identificador del proyecto al que está asociado este diagrama
Use Case	IdUseCase	Descripción de la solución de una atención
	InfRelacion	Información relativa a sus relaciones
BuildSecuence	IdBuildSecuence	Identificador de la secuencia de construcción generada

A.3. Principales clases desarrolladas:

Se tiene, a continuación, un listado de las principales clases desarrolladas, así como sus atributos más representativos.

a) Crear preguntas por caso de uso.

Se crearon las entidades: TUPUX.Entity. UMLQuestion.cs y TUPUX.Entity. UMLQuestionCollection.cs. Para asignar y obtener valores.

Propiedades: Id, Type y Description.

Métodos: GetQuestrionOptions.

Se modificó el archivo: TUPUX.Main. profiles. specification.TUPUX.Specification.prf.

Para poder guardar los valores a ingresar en los objetos del StarUml.

Estereotipo: Question, de tipo: UMLClass.

Definición de Tags: id, type y description.

Icono: questions.bmp.

Se agregaron las pestaña: Questions, al formulario: TUPUX.Forms.UseCaseEdit.cs.

Para poder guardar los valores a través de una interfaz gráfica.

Métodos: New, Save, Edit, Cancel, Remove y GetQuestions.

b) Crear opciones o respuestas por pregunta.

Se crearon las entidades: TUPUX.Entity. UMLQuestionOption.cs y TUPUX.Entity. UMLQuestionOptionCollection.cs. Para asignar y obtener valores.

Propiedades: Id, QuestionId y Value.

Se modifica el archivo: TUPUX.Main.profiles.specification.TUPUX.Specification.prf.

Para poder guardar los valores a ingresar en los objetos del StarUml.

Estereotipo: QuestionOption, de tipo: UMLAtribute.

Definición de Tags: id, questionid y value.

Se agrega el formulario: TUPUX.Forms.QuestionEdit.cs.

Para poder editar las preguntas y guardar los valores de las respuestas a través de una interfaz gráfica.

Métodos: New, Save, Edit, Cancel, Remove.

c) Crear mantenimiento de tipos de usuario.

Se crearon las entidades: TUPUX.Entity.UMLUserType.cs, TUPUX.Entity.UMLUserTypeCollection.cs y TUPUX.Entity.HelperInquests.cs.

Para asignar y obtener valores.

Propiedades: Id, Name, Value.

Métodos: GetUserTypes y WriteUserTypesInProfile.

Se agrega el menú: Tools/Inquests/UserType Settings, modificando el archivo: TUPUX.Main.TUPUX.EstimationAddIn.mnu.

Se agrega el script: TUPUX.Main.TUPUX.userTypeSettings.vbs.

Para abrir el formulario cuando seleccionemos la opción desde el menú.

Se agrega el formulario: TUPUX.Forms.UserTypeEdit.cs.

Para poder guardar los valores a través de una interfaz gráfica.

Métodos: New, Save, Edit, Cancel, Remove y LoadUserTypes.

d) Crear encuestas con sus respectivas respuestas.

Se crearon las entidades: TUPUX.Entity. UMLInquest.cs, TUPUX.Entity.UMLInquestCollection.cs, UMLInquestAnswer.cs y UMLInquestAnswerCollection.cs.

Para asignar y obtener valores.

Propiedades: Id, UserTypeId, UserName, Description y Value; QuestionId, QuestionOptionId y Value.

Métodos: GetInquestAnswers.

Creamos el archivo: TUPUX.Main.profiles.inquest.TUPUX.Inquest.prf.

Para poder guardar los valores a ingresar en los objetos del StarUml.

Estereotipos: Inquest, de tipo: UMLClass y InquestAnswer, de tipo; UMLAttribute.

Definición de Tags: id, usertypeid, username, description y value; questionid, questionoptionid y value.

Icono: inquests.bmp.

Se agrega el menú: Tools/Inquests/New Inquest, modificando el archivo: TUPUX.Main.TUPUX.EstimationAddIn.mnu.

Se agrega el script: TUPUX.Main.TUPUX.inquest.vbs.

Para abrir el formulario cuando seleccionemos la opción desde el menú.

Se agrega el formulario: TUPUX.Forms.InquestEdit.cs.

Para poder guardar los valores a través de una interfaz gráfica.

Métodos: Save, Clean, Cancel, Remove, LoadQuestion, LoadUserTypes, CreatePackageInquest y ValidInquest.

Bibliografía

[1] Booch, Grady; Rumbaugh, James y Jacobson, Ivar: The Unified Modeling Language User Guide; Primera Edición; 512 páginas; 1998.

[2] Pow-Sang, J.A., Nakasone, A., Imbert, R., Moreno, A.M.: An Approach to Determine Software Requirement Construction Sequences based on Use Cases, Proceedings Advanced Software Engineering and Its Applications-ASEA 2008 (Sanya, China), IEEE Computer Society, 2008

[3] StarUML - The Open Source UML/MDA Platform en http://staruml.sourceforge.net/en

[4] Ruhe, Greer: Quantitative Studies in Software Release Planning under Risk and Resource Constraints, Proceedings of the 2003 (ISESE'03)

[5] Cueva Lovelle, Juan Manuel: La complejidad del desarrollo de software, Universidad de Oviedo, Programa de Doctorado, 2001 en http://www.di.uniovi.es/~cueva/asignaturas/doctorado/2001/complejidad.pdf

[6] The design Structure Matrix en http://www.dsmweb.org/

[7] David Balbín, Michael Ocrospoma, Emanuel Soto y José Antonio Pow-Sang, TUPUX: An Estimation Tool for Incremental Software Development Projects, Departamento de Ingeniería, PUCP.

[8] Mirian Díaz: Comparación entre la metodología Rational Unified Process(RUP) y la metodología Extreme Programming (XP); Escuela de Ingeniería de Sistemas, INFOFIA 49; USMP en http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos.html

[9] Gengshen Du, Jim McElroy, and Guenther Ruhe: A Family of Empirical Studies to Compare Informal and Optimization-based Planning of Software Releases, Laboratory for Software Engineering Decision Support, University of Calgary

[10] Gengshen Du, Jim McElroy, and Guenther Ruhe, Ad Hoc Versus Systematic Planning of Software Releases – A Three-Staged Experiment, Laboratory for Software Engineering Decision Support, University of Calgary

[11] Günther Ruhe and Moshood Omolade Saliu: The Art and Science of Software ReleasePlanning, University of Calgary, IEEE Software, 2005

[12] Rosenberg, D., Scott, K.: Use Case Driven Object Modeling with UML. Addison-Wesley, Massachusets (1999)

[13] Pow-Sang, J.A.: La Especificación de Requisitos con Casos de Uso: Buenas y Malas Prácticas, II SISOFT 2003, Pontificia Universidad Católica del Perú, Lima - Perú, 2003

[14] Jaaksi, A.: A Method for Your Object-Oriented Project, Journal of Object-Oriented Programming, Vol 10. No 9, 1998.

[15] IEEE Recommended Practice for Software Requirements Specification IEEE Std 830-1998.