universidad del país vasco

euskal herriko unibertsitatea

Facultad de Informática **Informatika Fakultatea**

TITULACIÓN: Ingeniería Técnica en Informática d Sistemas

WebDiagram3.0: Incorporación de un prototipo de diálogo funcional mediante una metodología ágil.

Alumno/a: D./Dña. Julen Salgado Tomas

Director/a: D./Dña. Begoña Losada Pereda

Proyecto Fin de Carrera, julio de 2012

Resumen

En este proyecto de Fin de Carrera se ha creado una nueva versión de la herramienta *WebDiagram*, cuya función es facilitar el desarrollo de aplicaciones interactivas mediante la metodología *InterMod*.

La nueva versión de la herramienta se basa en la versión anterior y se diferencia en que contiene un prototipo del *Modelo de Diálogo* completamente funcional. Además, WebDiagram 3.0 incorpora la concurrencia y mejora algunos aspectos del diagrama.

En *WebDiagram2.0* se detectaron diversos fallos en la ejecución del programa de prototipado, difíciles de solucionar, que venían arrastrándose a lo largo de las versiones efectuadas de esta herramienta. Con el objetivo de conseguir eliminar todos estos bugs se decidió partir de cero e incorporar el desarrollo guiado por pruebas, también conocido como *Test-driven development*. Este proceso de desarrollo facilita la corrección de errores, primero escribiendo las pruebas que fallan, y después implementando el código que hace que pase la prueba satisfactoriamente.

Las pruebas han sido organizadas en base a *Objetivos de Usuario.* A lo largo de las iteraciones del proceso se han validado a nivel de requerimientos, presentación y funcionalidad, tal y como aconseja la metodología *InterMod.*

Finalmente, cabe destacar que se ha intentado llevar a cabo el proyecto haciendo un seguimiento activo junto con la directora y otro miembro evaluador para, principalmente, fomentar el trabajo en equipo.

Agradecimientos

Gracias a todas las personas que han hecho posible que haya finalizado el proyecto. A Begoña Losada como directora del proyecto por haber encaminado el proyecto en cualquier situación y a Juan Miguel López por haber aportado gran variedad de soluciones prácticas a los problemas. A todos mis amigos del pueblo que siempre me han preguntado por el estado de mi proyecto y a todos aquellos que me acompañaron o me acompañarán en clase.

A toda mi familia por haber estado encima y porque sé que siempre estarán ahí. Especialmente a mis padres Enrique y Marga, por haberme dejado dedicarme a lo que me gusta y a mi hermano Markel Salgado que tan mala suerte tuvo con las lesiones el año pasado en el primer equipo de fútbol de Eibar.

Finalmente, a todas las personas maravillosas que he conocido y a mi novia Nerea por haber tenido que aguantarme siempre que le hablaba de todo mi trabajo. ¡Gracias!

Índice

Índice de ilustraciones

Índice de tablas

1. Introducción

En este capítulo se presenta el contexto en el que se ha planteado este proyecto. Aunque el proyecto *WebDiagram* se haya creado a partir de la herramienta *Diagram*, se expondrán los antecedentes de *WebDiagram2.0*, dado que es la herramienta en la que se ha basado este proyecto, eliminando así cualquier dependencia hacia su antecesor, *Diagram*. Y para finalizar este capítulo, se muestra una guía general de cómo está organizado el documento.

1.1 Contexto

Hoy en día, cada vez se valora más la disponibilidad de una herramienta que se amolde a las necesidades de los usuarios. En la mayoría de los casos, los diseñadores y desarrolladores no tienen en cuenta las opiniones de los usuarios hasta llegar a la fase de evaluación, al final del proceso de desarrollo.

InterMod [1]es una metodología centrada en el usuario, con características ágiles y dirigida por modelos, desarrollada para el diseño de aplicaciones interactivas. Propone utilizar un diseño centrado en el usuario para definir los requisitos tanto funcionales, como no funcionales, además de describir la interacción persona-computador y evaluar los prototipos.

De este modo, tanto los desarrolladores como los usuarios tendrán una idea de cuál será el resultado final del programa, aun estando en una fase temprana. La herramienta *WebDiagram* facilita la creación de estos prototipos incorporando algunos de los modelos que propone la metodología *InterMod*.

1.2 Antecedentes

1.2.1 Metodología *InterMod*

InterMod[2] surgió como una metodología iterativa centrada en el usuario, simple y coherente. Propone un diseño centrado en el usuario para definir requisitos, describir los diálogos entre persona y computador y evaluar los prototipos. Además propone realizar una serie de modelos formales y prototipos intuitivos para la corrección de posibles errores de organización en el desarrollo de la interfaz.

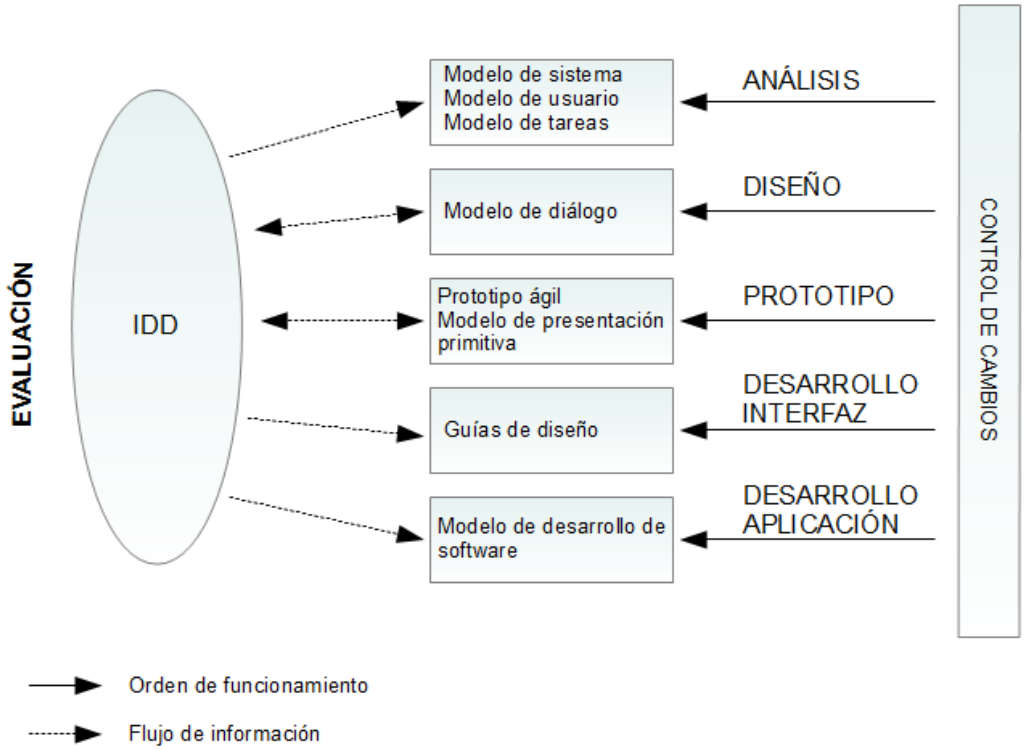


Ilustración 1: Proceso de InterMod

Se observa en la Ilustración 1 las fases que originan prototipos o modelos diferentes, y se comunican gracias a la *IDD (Intermediate Description Design)*. Los modelos se guardan en un nivel entre la conexión de las fases. Guardar la información facilita la recuperación del producto final y permite hacer todos los cambios deseados al prototipo principal. Cada prototipo obtenido y evaluado, nos informa sobre el nivel de desarrollo de la aplicación.

InterMod propone la realización de una serie de modelos:

- **Modelo de usuario**, especifica las características, límites y preferencias del funcionamiento del usuario.
- **Modelo de sistema**, recoge las características y limitaciones del Sistema.
- **Modelo de tareas**, define las acciones del usuario y su temporalidad en la consecución de los objetivos de usuario.
- **Modelo de diálogo**, es una evolución del modelo de tareas y maneja el resultado de la interacción del sistema. Incluye un modelo de Prototipado con características gráficas simples y un Modelo de Comportamiento que define la semántica de la aplicación.
- **Modelo de presentación**, define las características y componentes gráficos de la interfaz de la aplicación.
- **Modelos de Desarrollo de Software**, engloba los modelos clásicos de la lógica de negocio como los diagramas propios de UML, por ejemplo.

El Modelo de Usuario y el Modelo del Sistema influencian la configuración del Modelo de Tareas. El Modelo de Diálogo es una extensión del Modelo de Tareas. Tanto el Modelo de presentación como el Modelo de Desarrollo de Software recogen información del Modelo de Diálogo.

1.2.2 Evolución en la metodología *InterMod*

Una evolución posterior de la metodología InterMod [1] engloba tres filosofías: Model-driven Development, User-Centered Design y los métodos ágiles.

La incorporación de InterMod dentro de los métodos ágiles se justifica por la realización según Objetivos de Usuario (UOs). Los UOs nuevos surgen como nuevos deseos de usuario (UOs Directos) o como resultado de decisiones de división o fusión de UOs previos en cualquier momento del proceso (UOs Indirectos). En la Ilustración 3 se observa un esquema simplificado del proceso. El proceso InterMod comienza con el paso 0 (Analizar el Proyecto en General) al inicio del proyecto, y después continúa con una secuencia de iteraciones. Cada iteración incluye tres pasos, (i se refiere a la iteración i-ésima):

- Paso 1.i (Construir la Lista de Objetivos de Usuario)
- Paso 2.i (Planificar la Iteración Paralela)
- Paso 3.i (Realizar las Actividades de la Iteración).

Todas las iteraciones están guiadas por el mismo plan de actuación que divide el trabajo en diferentes actividades de los Objetivos de Usuario (UOs). Cada actividad está dirigida por modelos, y existen diferentes actividades de desarrollo e integración posibles en cada iteración, efectuadas por diferentes equipos de trabajo.

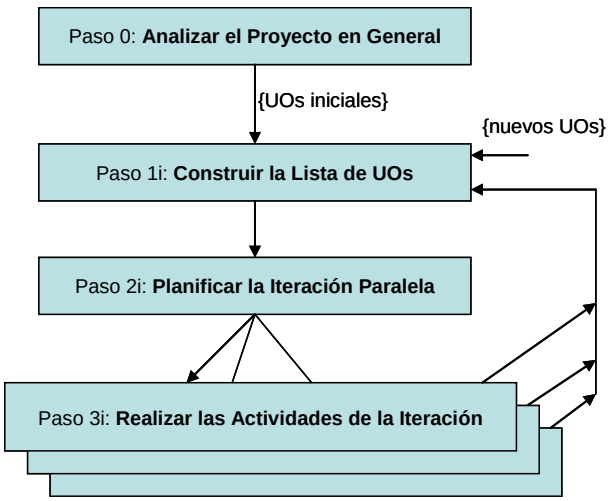


Ilustración 2: Proceso simplificado de InterMod

Por otro lado, se ha creado la herramienta *Diagram*, como ayuda para realización de la fase de análisis y diseño de la metodología *InterMod*. Para ello, transforma los modelos a formato *XML*.

1.2.3 Herramienta *Diagram*

Ha habido una evolución grande desde la primera versión de Diagram en 2007 hasta la versión que se presenta ahora. A lo largo de varios proyectos de fin de carrera, esta herramienta ha estado en constante desarrollo. En las nuevas versiones se han añadido nuevas funcionalidades y mejoras para la aplicación.

La herramienta surge en 2007, año en el que se desarrollan tres proyectos en paralelo:

- "*LOGRAM: Evaluador de webs en uso según los objetivos del diseñador*", en Junio de 2007; realizado por D. Francisco Villar, actualmente en desuso.

- "*EVALGRAM: una herramienta de ingeniería de interfaz para la evaluación temprana de interfaces interactivas con usuarios potenciales*", realizado por D. López, actualmente en desuso.

- "*Diseño e implementación de DIAGRAM: una herramienta para el modelado y evaluación de tareas de usuario*", desarrollado por J. Martínez Perdiguero. Esta primera versión de Diagram se ocupa del diseño del modelo de tareas de usuario y asimismo, dispone de una herramienta de prototipado simple.

Un año después, en junio de 2008, se presentan dos proyectos:

- "*Diseño centrado en el usuario de una web para una empresa de producción*" realizado por D. Ekaitz Diaz. Este proyecto tiene por objetivo la utilización de DIAGRAM 1.0 en el diseño de una web de empresa. La validación fue optimista.

- "*Realización de una herramienta de prototipo temprano de una aplicación interactiva a partir del diseño de tareas*", realizado por Borja Moreno. En este momento nace Diagram 2.0. Los prototipos son más avanzados, con forma de botones, base de los que se utilizan en las versiones actuales.

Más tarde, en 2009, se realizó el trabajo "*Diagram 3.0: Una herramienta de prototipado de aplicaciones interactivas según un modelo formal de tareas*", de Ainhoa de Sebastián. En esta versión aparece el modelo de tareas del Sistema y se añade un pequeño modelo de presentación (ventanas, secciones, botones desplegables). También se incluye un modelo de comportamiento simple con la opción de dirigir la ejecución mediante instrucciones "ir a" .

Pablo Garitano efectuó en 2010 "*WebDIAGRAM 1.0: Adaptación Web de DIAGRAM y puesta a punto del sistema*", que permitió llevar DIAGRAM a la Web.

En 2011 fueron defendidos cuatro proyectos relacionados con la herramienta:

- "*DiagramCMED: una herramienta para el desarrollo de software interactivo con visualización mediante mapas conceptuales*", realizado por Alberto Poncelas. Esta versión realizó con éxito la integración de la herramienta de mapas conceptuales CMED con WebDiagram 1.0. Esto permite realizar diagramas en forma de árbol más flexibles en los cambios y posicionamiento de sus nodos.

- *"Análisis y diseño de modelos de sistema y usuario para la herramienta Diagram"*, realizado por Jorge García, realizó la integración de los modelos del Usuario y del Sistema que recogen algunas características y limitaciones del usuario y del sistema, en este caso, en base a tres niveles: simple, medio y experto. De momento, esta herramienta no se ha propagado en las siguientes versiones.

- *"Método sistemático de verificación para aplicaciones software. Un caso a estudio con WebDiagram, herramienta de desarrollo de software interactivo"* realizado por Sofía Medina, realiza un estudio para definir un método sistemático de trabajo que guíe futuros desarrollos.

- "*Validación sémática de WebDiagram. Estudio práctica de herramientas de prueba*" realizado por Victor Ávila y cuyo objetivo es el análisis de herramientas de pruebas y detección de errores destinados a WebDiagram.

- "*WebDiagram 2.0: Zeregin banatuen eginkizuna*" de Lorea Ibáñez. Incorpora la delegación de tareas a Diagram.

1.3 Nuestra propuesta

Se quiere implementar una versión completamente funcional del modelo de diálogo de *WebDiagram*. En antiguas versiones del programa, los errores en la ejecución de ciertas tareas arruinaban el correcto funcionamiento de la herramienta.

Por ello se contempla la posibilidad de crear un nuevo prototipo de diálogo desde cero, e ir completando los objetivos establecidos uno a uno para evitar generar los mismos errores que su antecesor.

El funcionamiento será el mismo, dado que se implementará sobre la versión 2.0 del programa, aunque esto no implica dejar de lado posibles mejoras en la lógica de negocio. En la ilustración 3 se muestra un diagrama y su prototipo realizado con la versión WebDiagram2.0.

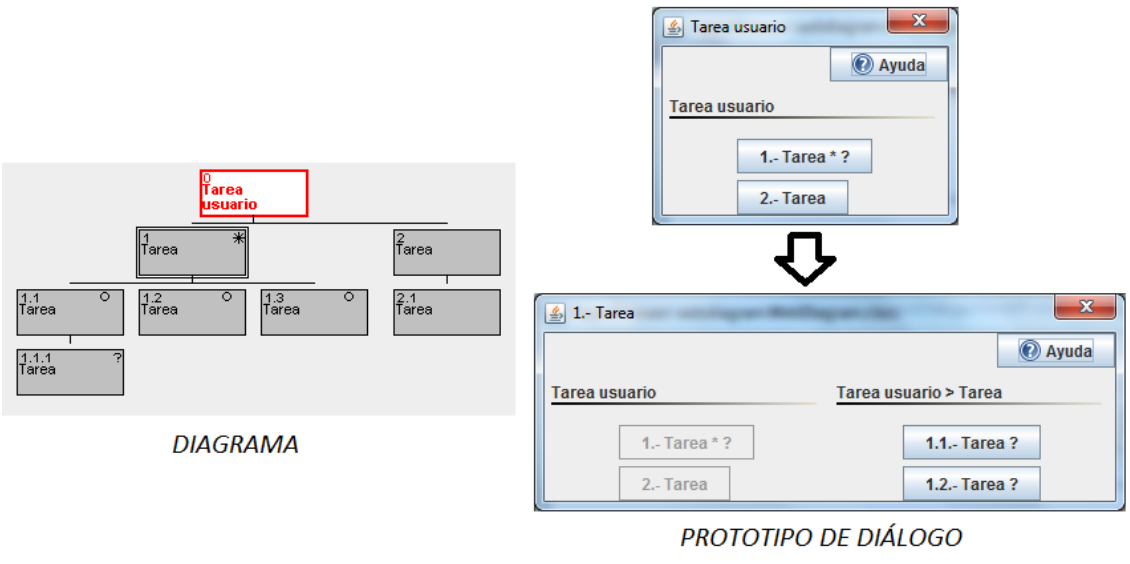


Ilustración 3: Diagrama y prototipo de diálogo de WebDiagram 2.0

En la realización de la nueva versión, se utilizarán las mismas herramientas de desarrollo, pero cambiará la metodología de desarrollo. En este caso se opta por la utilización de Test-Driven Development[3] y las propuestas de la metodología InterMod para asegurar un software de calidad.

1.4 Organización del documento

La estructura de esta memoria comienza, a partir de este punto, con el documento de objetivos del proyecto que se presenta en el capítulo 2 *"Documento de objetivos del proyecto"*. En él se muestra la motivación que ha llevado a realizar el proyecto, así como una pequeña descripción del mismo. De la misma forma, se presentan los objetivos a alcanzar, las fases y tareas a realizar, el análisis de riesgos (que incluye el plan de contingencias para cada riesgo detectado) y, por último, un análisis de factibilidad del proyecto.

En el capítulo 3 *"Planificación y gestión del proyecto"* se presenta la metodología usada para la gestión del proyecto. Se muestra el *Sprint Backlog* inicial que presenta en una tabla la estimación inicial realizada para cada tarea. Y a continuación se presentan, uno a uno, todos los *Sprint Backlogs* donde se pueden observar los resultados de las evaluaciones, tabla de estimaciones y decisiones tomadas en las reuniones.

En la siguiente sección, capítulo 4 *"La visión global"* se recoge la captura de requisitos tanto funcionales como no-funcionales y se realiza su correspondiente explicación. También se muestra la descripción general del funcionamiento del prototipo de diálogo mediante un ejemplo. Después se incluye el diagrama de clases que han sido usadas o modificadas y algunos detalles sobre los cambios realizados sobre el código del programa antecesor.

A continuación, en el capítulo 5 *"Conclusiones y líneas futuras"* se analizan los objetivos del proyecto para determinar si estos se han cumplido. Además, se incluyen algunas conclusiones tanto generales como personales surgidas durante la realización del proyecto y tras su finalización. Finalmente, se sugieren algunas posibles mejoras que se podrían realizar al programa en futuros proyectos.

El último capítulo es el correspondiente a las *"Referencias"* en el que se muestran las fuentes consultadas para recabar la información necesaria para la realización del proyecto y de la presente memoria.

Y para finalizar esta memoria, se incluyen anexos, los cuales contienen una muestra de las pruebas realizadas con los evaluadores.

2. Documento de objetivos del proyecto

En este capítulo se expone la motivación que ha llevado a realizar este Proyecto de Fin de Carrera, una descripción general del mismo y la metodología usada para llevar a cabo todas las fases y tareas del proyecto. También se hará una evaluación exhaustiva de cada riesgo con su respectivo plan de contingencias. Finalmente, se incluye el análisis de factibilidad en el que se valora si la realización del proyecto es posible.

2.1 Motivación

2.1.1 Aplicación

La metodología ágil *InterMod* y la herramienta *WebDiagram* ofrecen la oportunidad de mejorar o facilitar el trabajo de los diseñadores. De este modo, se obtendrán aplicaciones con un funcionamiento más fácil de entender desde el punto de vista del usuario.

En la actualidad, el desarrollo de aplicaciones interactivas conlleva una relación estrecha entre el diseñador y el usuario, característica imprescindible en metodologías ágiles como *InterMod*. De este modo, el usuario será capaz de corregir o comentar fallos con el diseñador en fases tempranas del proyecto. Y finalmente se consigue un producto o aplicación bien adaptada a las necesidades del usuario.

2.1.2 Personal

Personalmente, debo admitir que se me presenta un gran reto y oportunidad llevando a cabo un proyecto de tal magnitud. El no haber trabajado nunca antes en un proyecto ya en proceso de desarrollo, requiere cierta formación y cooperación antes de ponerse manos a la obra.

Por si esto fuera poco y con la motivación que esto requiere, poner en práctica todo lo que he aprendido en estos últimos años es lo que más me mueve. Trabajar en un pequeño grupo de desarrollo por un objetivo común (en este caso, *WebDiagram*), haciendo uso de herramientas y metodologías de gestión de proyectos es todo un golpe positivo para mi experiencia y formación como futuro ingeniero técnico en informática.

Resumiendo y teniendo en cuenta mis gustos en lo referente a la informática, el acierto en la selección de este Proyecto de Fin de Carrera es, en mi opinión, más que suficiente para lograr sacar de mí la motivación necesaria para poder llevar a cabo este trabajo.

2.2 Objetivos del proyecto

WebDiagram 3.0 es una versión web de Diagram que incorpora un prototipo del Modelo de Diálogo robusto y funcional, realizado con la metodología InterMod combinada con Test-Driven Development. Para ello se han determinado los siguientes objetivos:

- **Objetivo 1: Lograr el correcto funcionamiento de un prototipo de diálogo siguiendo un desarrollo guiado por pruebas.**
  Se crearán casos de prueba para cada iteración o *Sprint*, y se pondrán a prueba para ver los errores que genera la aplicación. A continuación se procederá a implementar parte del prototipo de diálogo para corregir dichos errores.
- **Objetivo 2: Gestionar el proyecto mediante el uso de una metodología ágil.**
  La gestión y planificación del proyecto se realizará siguiendo las reglas de una variación de la metodología ágil *Scrum*.
- **Objetivo 3: Reparar y mejorar los aspectos de WebDiagram2.0.**
  Se pretende corregir todos los fallos detectados en la versión anterior de la herramienta detectándolos mediante una sucesión de batallas de pruebas.

2.3 Fases y tareas del proyecto

En esta sección se presentan las distintas fases del proyecto así como las tareas que se realizarán en cada una de ellas:

2.3.1 Formación

Teniendo en cuenta los conocimientos que se cree que hacen falta para la realización de este proyecto, la formación se centrará principalmente en los siguientes temas:

- **Estudio de la documentación completa de *WebDiagram 2.0*.** Se realizará un estudio profundo del código y la memoria de la herramienta para poder entender completamente el funcionamiento de la misma, ya que el proyecto se centra básicamente en la modificación parcial del programa.
- **Estudio de las librerías gráficas de *Java*.** Se procederá a estudiar el uso de la librería *SWING* basada en *SWT* para la construcción de la interfaz de la aplicación y se requerirá manejarlo con cierta soltura para evitar problemas de diseño.
- **Investigación en patrones *Java*.** Será necesario documentarse sobre ciertos patrones usados en el desarrollo de la herramienta, así como sobre el patrón *Observer* o *Singleton*.
- **Investigación de las metodologías ágiles.** Dado que el proyecto entero se considera como parte de la formación de las metodologías ágiles, será necesaria una pequeña investigación anterior para obtener las nociones básicas y emprender el proyecto sin ningún problema.

No se tiene en cuenta la formación en herramientas ajenas al propósito del proyecto, dado que se considera que no se necesita estudio adicional en ellas, como *Microsoft Word* (para el desarrollo de los documentos), *LibreOffice Draw* o *Dia* (siendo estas dos últimas, herramientas para el diseño de diagramas de flujo).

2.3.2 Instalación del entorno de trabajo

En esta fase se procederá a la instalación de todas las herramientas de trabajo necesarias para llevar a cabo el proyecto, entre ellas:

- **Eclipse versión 3.7, Indigo:** tanto para la programación en Java como para las pruebas de *WebDiagram* y para el estudio del código de las anteriores versiones de la aplicación.
- **Microsoft Office Word 2010:** para completar la escritura de todos los documentos necesarios y la memoria del proyecto.
- **Microsoft Office Excel 2010:** para manejar tablas y cálculos de las horas estimadas para el proyecto, o el registro de horas invertidas.
- **LibreOffice Draw 3.4:** para construir los diagramas de flujo.

2.3.3 Gestión

La gestión del proyecto se llevará a cabo conforme a las siguientes tareas:

- **Reuniones:** se exigirá la participación en diversas reuniones a lo largo del desarrollo del proyecto. En ellas, se acordarán las soluciones y alternativas más efectivas para avanzar en el proyecto. Ya sean sobre la gestión de éste, el ajuste de los plazos de entrega o el análisis y evaluación de *WebDiagram* para asegurar de que todo va según lo previsto. Se decide establecer reuniones semanales, salvo en períodos de vacaciones y exámenes.
- **Realización de la planificación inicial:** el desarrollo del proyecto partirá de una planificación inicial que tendrá como fruto, completar el primer objetivo. Al final del proyecto, se mostrará la planificación real.
- **Aplicar una metodología ágil para la gestión del proyecto:** se hará uso de una metodología ágil para gestionar un avance progresivo en el proyecto. En este caso, la metodología será una variación de Scrum. En cada iteración, se evaluará lo realizado y lo pendiente para decidir el trabajo a realizar en la próxima iteración. De acuerdo a esto, se realizarán nuevas estimaciones sobre la planificación anterior.
- **Registro y documentación del proyecto:** se llevará un registro actualizado paulatinamente de todas las tareas completadas y en progreso, junto con las horas totales invertidas en el proyecto. El mantenimiento de la documentación se hará de manera gradual en cada fase del proyecto.
- **Copias de seguridad:** regularmente se harán copias de seguridad del archivo completo para minimizar las desastrosas consecuencias que causarían la pérdida de datos parcial o completa.

2.3.4 Desarrollo del modelo de diálogo de *WebDiagram*

Se realizará un desarrollo incremental e iterativo del modelo de diálogo, partiendo desde cero. Los Objetivos de Usuario (UO) de partida son los siguientes:

UO1: Representar tareas de Orden Secuencial

UO2: Representar Tareas de Orden Secuencial y Elección

UO3: Representar tareas de Orden Secuencial, Elección e Indiferente

UO4: Representar tareas de Orden Secuencial, Elección e Indiferente con diferentes modelos de presentación

UO5: Incorporar la representación de tareas del Sistema

UO6: Incorporar el modelo del Comportamiento

2.3.5 Preparación de los casos de prueba

Antes de cada iteración será necesario disponer y actualizar los casos de prueba, asociados a los Objetivos de Usuario, que servirán para evaluar el producto software disponible.

2.3.6 Documentación

Aunque el desarrollo de la documentación se realice desde el inicio del proyecto (al final de cada iteración), esta fase se dedicará a completar y corregir la documentación del proyecto.

- **Creación de la memoria:** se completará la memoria que ha sido desarrollada poco a poco a lo largo de todo el proyecto.
- **Preparación de todo el material para la defensa del proyecto:** cuando se haya completado la memoria, se desarrollará todo el material necesario para realizar la defensa del proyecto.
- **Defensa del proyecto:** al finalizar el proyecto, se procederá a la presentación del mismo ante un tribunal.

2.4 Análisis de riesgos

El hecho de que estemos ante un proyecto de gran magnitud, al igual que todos los trabajos de esta categoría es necesario realizar un análisis de los riesgos que puedan poner en peligro el curso del proyecto. En él, se deberán identificar bien los riesgos y las consecuencias que puedan ocasionar si dichos riesgos ocurrieran. Por ello, también se elaborará un plan de contingencia para cada riesgo.

En la siguiente tabla se han recogido los riesgos junto con sus respectivas consecuencias y planes de contingencia. Los riesgos están clasificados en tres niveles: bajo, medio, alto. Con los que se representa la probabilidad con la que pueden ocurrir dichos riesgos. Al igual que el impacto puede ser débil, medio o fuerte dependiendo de la gravedad de las consecuencias que acarrea.

Finalmente, se presenta la forma de prevenir el riesgo mediante soluciones sencillas, que en caso de no ser efectivas, se procederá a la solución más efectiva; generalmente la inversión de más horas de trabajo.

Riesgo	Razón	Nivel de riesgo	Impacto	Prevención	Solución
Enfermedades	Contraer una grave enfermedad que impida avanzar	Bajo	Fuerte	-	Recuperar las horas perdidas en posteriores sesiones

Riesgo	Razón	Nivel de riesgo	Impacto	Prevención	Solución
Poca experiencia con SWING	Entornos poco trabajados	Bajo	Medio	Realizar una buena formación con antelación	Invertir más horas
Mala planificación	Por falta de experiencia en estimaciones	Alto	Medio	Planificar dejando siempre un margen de tiempo	Invertir más horas
Pérdida de datos	Pérdida parcial o completa del archivo del proyecto	Bajo	Fuerte	Realizar copias de seguridad regularmente	Invertir más horas
Carga de trabajo externo	Por el hecho de tener clases, trabajos y actividades extraescolares	Alto	Medio	Invertir más horas en el tiempo libre	Ajustar los plazos

Tabla 1: Análisis de riesgos

2.5 Análisis de factibilidad

Una vez evaluado el análisis de riesgos y los posibles problemas que podrían traer, se examina la disponibilidad de las herramientas y técnicas de los que se hará uso para lograr exitosamente los objetivos del proyecto.

El mayor riesgo se deriva de la falta de experiencia del autor en el campo de gestión de proyectos y programación de la librería gráfica de *Java*. Por esta razón, se ha decidido realizar una planificación con cierta holgura para tener un amplio margen en caso de que se necesite invertir más horas de trabajo. Para tomar esta decisión, también se ha tenido en cuenta el tiempo de la fase de formación, que podría variar según vaya avanzando el proyecto.

Finalmente, el coste económico que supondría el desarrollo del proyecto se ha determinado nulo, ya que se hace uso de herramientas gratuitas o las que ofrece la Facultad de Informática a todos los estudiantes sin coste alguno. Teniendo en cuenta todo ello, se deduce que se puede llevar a cabo el proyecto.

3. Planificación y gestión del proyecto

En los siguientes apartados se explicarán detalladamente la metodología usada para la gestión del proyecto y las fases que serán necesarias para hacerlo, junto con las especificaciones de cada una de ellas.

3.1 Metodologías ágiles

Las metodologías usuales definen un proceso secuencial, y cada proceso se basa en el proceso anterior. La mayoría de los documentos se desarrollan en la primera fase; por esta razón, el proyecto sigue un único plan durante todo su desarrollo.

Por otro lado, las metodologías ágiles usan una planificación adaptada para cada iteración determinada en un corto periodo de tiempo, con las que logramos una pequeña muestra de pruebas del software final. Por ello, un proyecto se puede ver como un grupo de pequeños proyectos de carga menor, donde la planificación puede variar y ser adaptada.

Estas metodologías tienen ciertas características que las han hecho especialmente atractivas para este proyecto:

- **Planificación adaptativa:** el proyecto se divide en varias iteraciones independientes. De este modo, se van adaptando las planificaciones de las fases sucesivas a medida que se contemplan las anteriores. Esto nos permite reducir el impacto de cualquier riesgo que pudiera materializarse.
- **Creación de software funcional en plazos cortos de tiempo:** al final de cada iteración se entrega una muestra del software producido. Así, el cliente puede ir viendo el avance del proyecto sin esperar a que esté completamente terminado. Evitando así la carga de trabajo que generaría cualquier modificación en la versión definitiva.
- **Comunicación con el cliente:** en estas metodologías se incluye al cliente como parte activa del desarrollo, lo que produce una mayor satisfacción por su parte en el resultado final. Se mantiene el contacto continuo con el cliente, quien aporta su opinión sobre el producto, minimizando así el impacto de cualquier cambio en los requisitos.
- **Escasa documentación:** se considera que la creación de un buen software es más productivo que el desarrollo de una buena documentación, ya que será el software lo que determinará el avance del proyecto.

Si bien los factores anteriores determinaban los principios básicos de las metodologías ágiles, los siguientes puntos muestran la diferencia con las metodologías usuales:

1. Satisfacer las necesidades del cliente será prioritario, y para ello a medida que avance el proyecto, se le entregaran modelos del software al cliente para que vea la evolución del proyecto.
2. Permitir cambios, para que el cliente tenga una ventaja competitiva.

3. Facilitar un software en un plazo de 2 a 8 semanas, siendo el plazo de entrega lo menor posible.
4. La colaboración de cualquier trabajador y desarrollador del proyecto será indispensable.
5. Selección de gente a gusto con el trabajo, proporcionarles un buen entorno y transmitir confianza en ellos.
6. Una reunión anticipada es la manera esencial de compartir toda la información.
7. El funcionamiento del software es la señal clara del avance en el proyecto.
8. Las metodologías ágiles promueven una gestión sostenible. Los promotores, desarrolladores y usuarios son capaces de mantener constantemente un ambiente de paz y tranquilidad.
9. Un buen diseño aumenta la ligereza y hay que tener atención continua sobre la calidad del proyecto.
10. La simplicidad es esencial.
11. Los mejores diseños y arquitecturas son producto de la buena organización del grupo.
12. De vez en cuando, el grupo reflexionará sobre cómo mejorar para lograr mayor productividad.

3.2 Metodología empleada

Existen prácticas ágiles que integran Scrum con otras metodología ágiles: Scrum + Evo, Scrum + UP, Scrum + XP, etc. En este PFC se utilizará Scrum junto con la metodología InterMod y TDD (Test-Driven Development). En la ilustración 5 se visualiza la metodología conjunta empleada en este trabajo. Estará basado parcialmente en la gestión de Scrum , el proceso de desarrollo estará guiado por InterMod y las evaluaciones se realizarán al comienzo de las iteraciones al estilo TDD. Se explica a continuación con más detalle.

Ilustración 4: Metodologías de trabajo y gestión empleadas conjuntamente en este trabajo

Se utilizará una variación de la metodología Scrum en lo referente a la gestión [Larman] para realizar las estimaciones de este proyecto. Se utilizará el término Sprint Backlog para la estimación en horas del trabajo restante para cada tarea.

El proceso del trabajo se hará siguiendo las directrices de la metodología InterMod, descrita brevemente en el apartado XX.

Las anteriores versiones de Diagram habían descubierto una gran complejidad en los requerimientos funcionales y numerosas interdependencias en ellos. Resulta difícil definir todos los requerimientos en su totalidad desde un principio, y la modificación posterior de unos requerimientos altera otras funcionalidades. Por este motivo, se decide:

- Incluir **el Desarrollo Guiado por Pruebas (*Test-driven Development*) en las evaluaciones y para guiar las iteraciones:** Se crearán pruebas (*Análisis y Diseño de la Navegación*) para los diferentes Objetivos de Usuario que vayan surgiendo y se validarán al inicio de cada iteración.

El proceso será el siguiente:

Al comienzo de cada iteración se realizarán las pruebas sobre el software disponible. Las evaluaciones se realizarán conjuntamente por un grupo multidisciplinar compuesto por el autor del presente PFC (implementador), Begoña Losada (especificación de pruebas), Juan Miguel López (evaluador). En los procesos de evaluación, se validan aspectos relativos a los requerimientos, aspectos de presentación y aspectos relativos a la implementación correcta de la lógica de negocio.

Tras la evaluación de las pruebas, se decidirán para esa iteración: 1. Los Objetivos de Usuario a desarrollar y 2. Las actividades de desarrollo e integración a realizar.

3.3 Planificación del proyecto. Sprint Backlog inicial

Este apartado sustituye al Diagrama de Gantt de las metodologías clásicas. En él se muestra una tabla donde se han especificado aproximadamente las horas necesarias para realizar cada una de las tareas. Además se adjunta una columna Objetivos, que permite relacionar las tareas con los objetivos ya predefinidos en el Documento de Objetivos del Proyecto:

- **O1:** Lograr el correcto funcionamiento de un prototipo de diálogo siguiendo un desarrollo guiado por pruebas.
- **O2:** Gestionar el proyecto mediante el uso de una metodología ágil.
- **O3:** Reparar y mejorar los aspectos de WebDiagram2.0.

Fases	Tareas		Horas totales estimadas	Objetivos
Gestión	Reuniones	Preparación del orden del día	1:00	O2
		Realizar la reunión	15:00	O2
		Recoger el acta	3:00	O2
	Planificación inicial		5:00	O2
	Realizar el Sprint Backlog		6:00	O2
	Archivo	Mantenimiento	1:00	O2

Fases	Tareas		Horas totales estimadas	Objetivos
		Copias de seguridad	2:00	O2
Formación	WebDiagram		4:00	O1, O3
	Librerías gráficas		6:00	O1, O3
	Patrones Java		2:00	O1, O3
	Metodologías ágiles		4:00	O2
Instalación	Instalación de las herramientas necesarias		2:00	O1, O2, O3
Captura de requisitos	Casos de uso		1:00	O2
	Prototipo de la interfaz		2:00	O3
Diseño	Diseño de la lógica de negocio		20:00	O1, O3
	Diseño del modelo de presentación		10:00	O1, O3
Implementación	Definir la arquitectura		1:00	O1, O3
	Implementación de la interfaz		3:00	O3
	Implementación de la lógica de negocio	Tareas de orden secuencial	6:00	O1, O3
		Tareas de elección	7:00	O1, O3
		Tareas de orden indiferente	10:00	O1, O3
		Tareas de orden concurrente	7:00	O1, O3
		Presentación	9:00	O1, O3
		Tareas del sistema	2:00	O1, O3
		Funcionalidad “Ir A”	6:00	O1, O3
Pruebas	Diseño de pruebas		10:00	O2
	Ejecución de pruebas		15:00	O1, O3
Documentación	Desarrollo de la memoria		60:00	O2
	Creación del manual de usuario		2:00	O2
	Elaboración de la presentación		15:00	O2
TOTAL			237:00	

Tabla 2: Sprint Backlog inicial

Como se puede observar en la Tabla 2, el total de horas previstas para la planificación inicial del proyecto es de 237 horas. Con lo cual, podemos concluir que en el caso de que se cumpliera esta previsión, el total de horas sobrepasaría la dedicación de aproximadamente 150 horas que requiere un proyecto de fin de carrera en la ingeniería técnica.

En el capítulo 8 se muestran las estimaciones de las horas que se han necesitado para cada iteración y una comparación de todas ellas para visualizar gráficamente la carga de trabajo que han supuesto.

3.4 Gestión del proyecto. Sprint Backlog sucesivos

En esta sección se presenta la gestión que se ha llevado a lo largo del proyecto. En él, se muestra la estimación del tiempo que se ha calculado para realizar todas las tareas en varias iteraciones.

Se decide realizar reuniones de evaluación cada semana. Sin embargo, este plazo es flexible y puede ampliarse o acortarse, dependiendo del esfuerzo necesario para obtener el subproducto o prototipo. También influyen en la duración de la iteración factores externos al proyecto como vacaciones o prioridad de otros trabajos. Cada iteración comienza con una reunión de evaluación en la que el equipo realiza las pruebas sobre el software existente. Y a continuación se efectúa lo siguiente:

- Se recogen los resultados de la evaluación. En algunos casos, tal y como aconseja TDD, se realizan los cambios directamente en la sesión de evaluación.
- Se realizan nuevas estimaciones, se ajusta la tabla de estimaciones según los resultados de la evaluación. Las columnas en la tabla expresan lo siguiente:
  - ✓ **Estimación anterior:** Se muestran las horas estimadas en la iteración anterior para realizar una tarea.
  - ✓ **Horas invertidas:** Se muestran las horas invertidas en la iteración anterior para completar una tarea.
  - ✓ **Nueva estimación:** Se estima un incremento/decremento en las horas restantes para la realización de la tarea. Por ejemplo, si se prevé que una tarea no podrá realizarse dentro de las horas estimadas, se procederá a aumentar la cantidad de horas.
- Se toman unas decisiones de diseño, de presentación y/o de implementación. Pueden ser nuevos Objetivos de Usuario o cambios sobre anteriores Objetivos.
- Se deciden los Objetivos y las actividades a realizar en la iteración.

1ª Sprint Backlog: Del 25/11/2011 al 15/12/2011

En esta primera iteración, se ha analizado los objetivos del proyecto y se ha establecido como primer objetivo lograr el correcto funcionamiento del prototipo de diálogo con las tareas secuenciales y unitarias. Dado que se acordó seguir el desarrollo guiado por pruebas (*Test-driven Development*), se ha decidido informarse acerca de esta metodología además de realizar un estudio profundo del anterior WebDiagram.

Para la creación del nuevo prototipo de diálogo, se ha propuesto eliminar la implementación actual del prototipo y desarrollar uno nuevo desde el principio. Se cree que es una buena decisión que evitará reaparecer problemas del antiguo prototipo.

Resultado de la evaluación

1. Para comenzar con la creación del prototipo de diálogo, se ha borrado completamente la implementación actual del mismo. Por esta razón, el prototipo de diálogo no inicia. En base a este fallo provocado, se han tomado las decisiones descritas más abajo.

Tabla de estimaciones

La tabla del primer *Sprint* no se muestra porque es exactamente igual que la que se ve en el *Sprint Backlog* inicial. La nueva estimación es la que se ha calculado al inicio del proyecto ya que todavía no se han invertido horas de trabajo.

Decisiones de diseño

- ✓ Hay que tener en cuenta que nunca habrá más de una ventana activa. A la hora de pulsar una tarea con hijas, se remplazará la ventana actual con la siguiente que contendrá las tareas hijas. Y al finalizar todas las tareas de una ventana, se volverá automáticamente a la ventana anterior, o se finalizará el prototipo si es la última ventana de la pila.

Decisiones de presentación

- ✓ Manteniendo el formato de las ventanas de la versión anterior, la ruta y el nombre de la tarea actual en ejecución se mostrará arriba, seguido de la secuencia de botones que representarán cada tarea hija.

Decisiones de implementación

- ✓ Para la implementación se ha decidido eliminar completamente el módulo referente al prototipo de diálogo y comenzar el desarrollo del mismo desde el principio. Se reducirá la cantidad de clases necesarias para su funcionamiento:
    - **GestorPrototipoDialogo.** Esta clase será la encargada de ejecutar o finalizar el prototipo de diálogo y gestionará las ventanas en una pila. Para ello, se valdrá de métodos que crearán y finalizarán ventanas según la situación.
    - **VentanaPrototipoDialogo.** Esta clase heredará la clase *JDialog* gestionando las tareas y los botones que se alojen en ella. Se encargará de capturar las acciones de cada botón y de decidir la acción correcta en cada caso. También comprobará si todas las tareas han sido ejecutadas para proceder a la finalización del prototipo usando el método del *GestorPrototipoDialogo*.
    - **Boton.** La clase *Boton* será una extensión de la clase *JButton* que guardará atributos extra para identificar la tarea a la que pertenece y las veces que se ha ejecutado.
    - **SubprogramasProrotitpoDialogo.** Se ha mantenido parcialmente la implementación de esta clase, dejando solamente 3 métodos, que son necesarios para el dibujo del nombre y la ruta de las tareas.

Decisiones de gestión

- ✓ El proceso de desarrollo se basará en la metodología *Test-driven Development*, donde las pruebas guiarán la implementación hasta conseguir el objetivo general ya establecido: **Implementación de un prototipo de diálogo funcional**.

Planificación de actividades para esta iteración

- Lograr el correcto funcionamiento en las tareas secuenciales pasando por todas las fases de la iteración.
- Completar el Documento de Objetivos del Proyecto y los apartados acerca de la gestión y planificación (realizar estimaciones).
- Estudio profundo de la implementación de WebDiagram 2.0.
- Decidir el idioma en el que se desarrollará el proyecto.

2ª Sprint Backlog: Del 16/12/2011 al 22/01/2012

En el segundo *Sprint*, se ha mejorado el prototipo de diálogo para que muestre un comportamiento adecuado con las tareas de orden secuencial que sean de cualquier tipo. Cabe destacar que se han tenido que tomar nuevas decisiones para mejorar la interacción con el usuario a la hora de usar tareas opcionales y repetitivas.

La implementación genérica realizada en la anterior iteración ha facilitado el desarrollo de este objetivo, ya que se tuvieron en cuenta los posibles cambios que habría que hacer para completar el prototipo con el funcionamiento de las tareas de tipo opcional y repetitivo.

Resultado de la evaluación

1. Implementando todas las clases indicadas anteriormente, se ha conseguido un prototipo de diálogo funcional con las tareas unitarias y de orden secuencial con el siguiente aspecto:

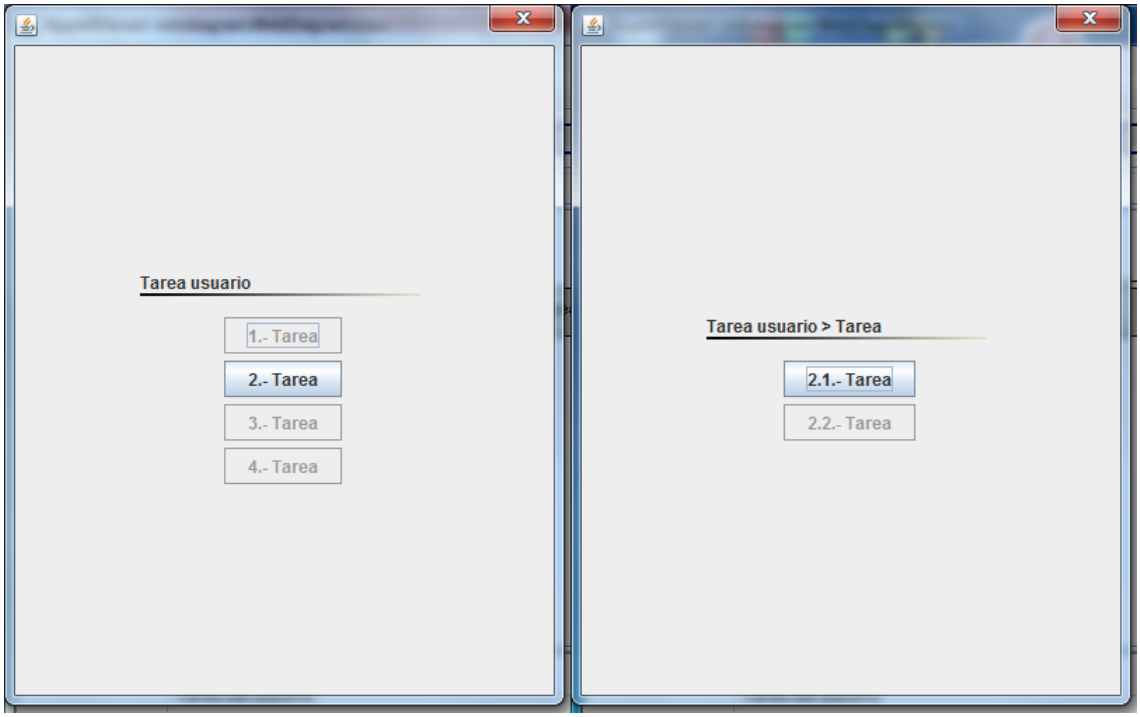


Ilustración 5: Prototipo de diálogo con tareas secuenciales y unitarias

2. Las tareas secuenciales de tipo opcional y repetitiva no funcionan, se comportan como si fueran unitarias. De modo que se crearán nuevas pruebas que guiarán el desarrollo del prototipo de diálogo para las tareas secuenciales de todos los tipos.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	1:00	0:05	0:55

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
		Realizar la reunión	15:00	2:30	12:30
		Recoger el acta	3:00	0:30	2:30
	Planificación inicial		5:00	2:30	2:30
	Realizar el Sprint Backlog		6:00	0:30	5:30
	Archivo	Mantenimiento	1:00	0:10	0:50
		Copias de seguridad	2:00	0:15	1:45
Formación	WebDiagram		4:00	4:00	0:00
	Librerías gráficas		6:00	1:00	5:00
	Patrones Java		2:00	0:45	1:15
	Metodologías ágiles		4:00	1:00	3:00
Instalación	Instalación de las herramientas necesarias		2:00	1:30	0:30
Captura de requisitos	Casos de uso		1:00	0:00	1:00
	Prototipo de la interfaz		2:00	0:30	1:30
Diseño	Diseño de la lógica de negocio		20:00	0:30	19:30
	Diseño del modelo de presentación		10:00	2:30	7:30
Implementación	Definir la arquitectura		1:00	0:45	0:00
	Implementación de la interfaz		3:00	0:30	2:30
	Implementación de la lógica de negocio	Tareas de orden secuencial	6:00	3:30	2:30
		Tareas de elección	7:00	0:00	7:00
		Tareas de orden indiferente	10:00	0:00	10:00
		Tareas de orden concurrente	7:00	0:00	7:00
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	6:00	0:00	6:00
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		15:00	0:10	14:50
Documentación	Desarrollo de la memoria		60:00	10:00	50:00
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 3: 2ª Sprint Backlog

Decisiones de diseño

✓ Se ha aceptado la nueva funcionalidad de cerrar las ventanas en caso de tener tareas opcionales. Cuando el prototipo se encuentra en el caso de la siguiente imagen, la decisión del usuario podría ser realizar la tarea opcional restante, o simplemente ignorarla.

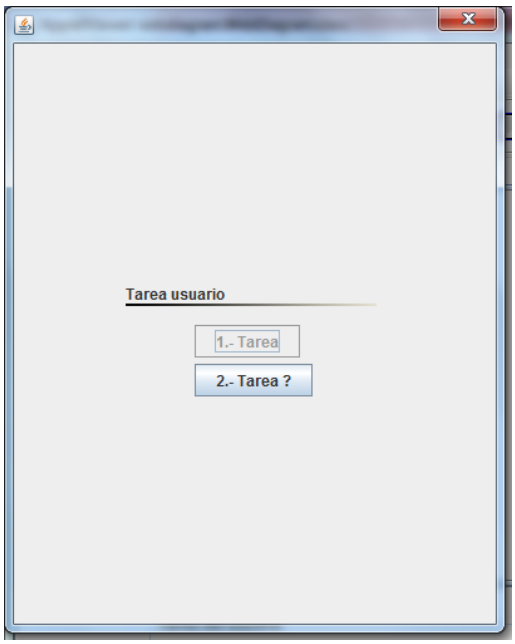


Ilustración 6: Tarea opcional del prototipo

Una solución a este dilema podría ser la creación de un botón "Finalizar" que se encargaría de finalizar la tarea en caso de que las tareas restantes fueran opcionales. Sin embargo, se ha pensado otra solución más interesante que incluso mejoraría la interacción con el usuario. Hablamos del uso del **botón de cierre** de la ventana.

Decisiones de implementación

✓ La reacción a la pulsación del **botón de cierre** sería distinta dependiendo del estado de las tareas de la ventana:

- Si se pulsara el botón sin haber terminado de ejecutar todas las tareas obligatorias, se cancelaría la ejecución del prototipo mostrando un mensaje de lo ocurrido.
- En caso de haber ejecutado todas las tareas obligatorias y haber dejado una o más tareas opcionales sin ejecutar, el prototipo finalizaría correctamente. Para puntualizar, si la ventana no es la última de la pila, ésta se cerrará y continuará la ejecución en la ventana anterior.

Planificación de actividades para esta iteración

- Seguir completando la memoria con los cambios realizados en la implementación y realizar las estimaciones necesarias para la siguiente iteración.
- Corregir errores conocidos en la lógica de negocio del programa antecesor que ya están documentados, ajenos al prototipo de diálogo.

- Al recibir las nuevas pruebas junto con los fallos que generen, se procederá a la implementación de las tareas secuenciales de tipo opcional y repetitiva. Mientras tanto se comenzará su desarrollo en base a las decisiones tomadas.
- Reflexionar sobre la dificultad de la implementación que supondrá la aprobación de algunas de las decisiones en un futuro, evitando así cambios bruscos en los requisitos.

3ª Sprint Backlog: Del 23/01/2012 al 29/01/2012

En este *Sprint* se pasará a completar la funcionalidad de la herramienta para que admita la ejecución de tareas de orden de elección, sea cual sea el tipo. Se centrará exclusivamente en el desarrollo de esta característica, dejando de lado los fallos del panel de propiedades que puedan perjudicar la ejecución. Para ello se supondrá que estos fallos están corregidos y no se realizarán pruebas que comprometan su estabilidad.

Esto no quiere decir que no se vaya a hacer nada al respecto, la corrección de los fallos del panel de propiedades se dejará para las iteraciones posteriores.

Resultado de la evaluación

1. Las pruebas han sido satisfactorias. Se ha encontrado un detalle que debe arreglarse en los botones de tareas opcionales, que se puede apreciar claramente en la siguiente imagen.

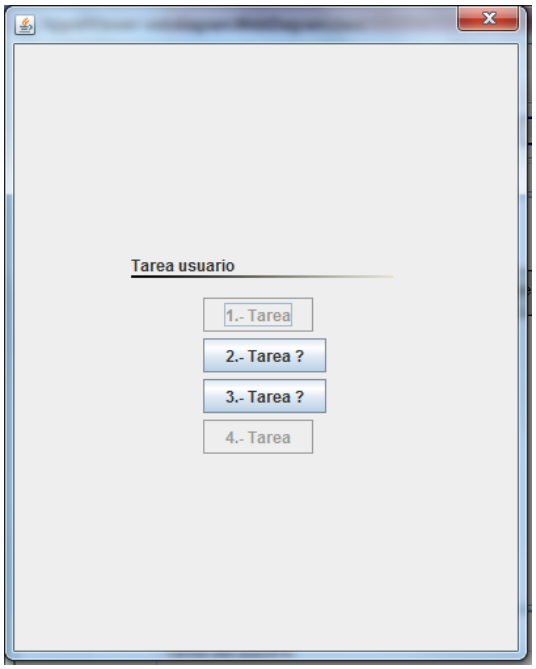


Ilustración 7: Fallo en las tareas opcionales

El problema reside en la visibilidad de los botones cuando una o más tareas opcionales siguen a otra, en este caso la 4. Tarea debería estar activada. Esto ocurre por una mala implementación que se encarga de activar el botón de una tarea cuando se detecta que la anterior es opcional, pero no se encarga de activar el botón cuando éste sucede a más de un botón opcional.

2. Fuera del funcionamiento del prototipo de diálogo, se ha visto que la casilla “IR A” del panel de propiedades no está activada si la 1. Tarea es seleccionada, pero no debería ser así.

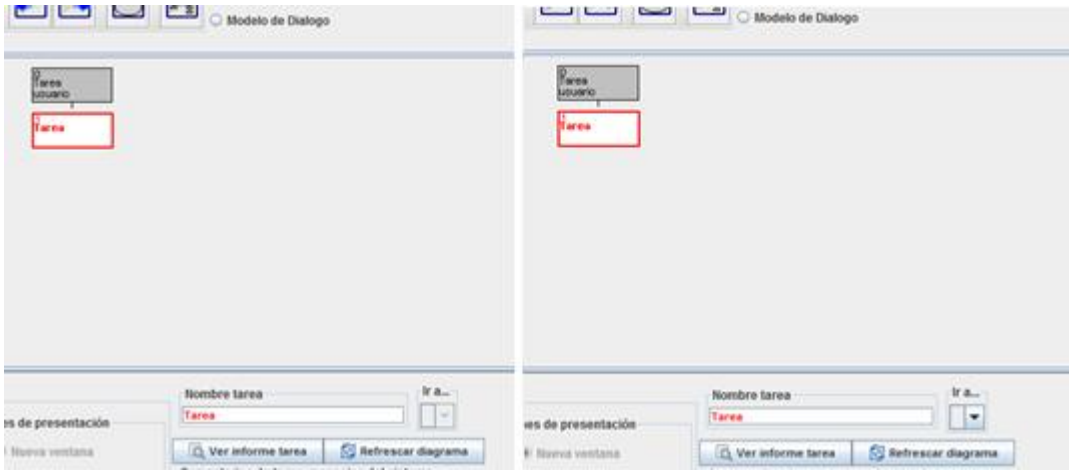


Ilustración 8: Visibilidad del IR A en la 1. Tarea

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:55	0:10	0:45
		Realizar la reunión	12:30	1:30	11:00
		Recoger el acta	2:30	0:30	2:00
	Planificación inicial		2:30	1:50	0:00
	Realizar el Sprint Backlog		5:30	1:00	4:30
	Archivo	Mantenimiento	0:50	0:00	0:50
		Copias de seguridad	1:45	0:05	1:40
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		5:00	0:00	5:00
	Patrones Java		1:15	0:00	1:15
	Metodologías ágiles		3:00	0:30	2:30
Instalación	Instalación de las herramientas necesarias		0:30	0:15	0:15
Captura de requisitos	Casos de uso		1:00	0:30	0:00
	Prototipo de la interfaz		1:30	0:00	1:30
Diseño	Diseño de la lógica de negocio		19:30	2:15	17:15
	Diseño del modelo de presentación		7:30	0:00	7:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		2:30	0:00	2:30
	Implementación	Tareas de	2:30	2:30	1:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	de la lógica de negocio	orden secuencial			
		Tareas de elección	7:00	0:00	7:00
		Tareas de orden indiferente	10:00	0:00	10:00
		Tareas de orden concurrente	7:00	0:00	7:00
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad "Ir A"	6:00	0:10	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		14:50	0:30	14:20
Documentación	Desarrollo de la memoria		50:00	0:45	49:15
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 4: 3ª Sprint Backlog

Decisiones de implementación

✓ Para corregir el error **(número 2)** detectado en las pruebas y así permitir que la primera tarea pueda tener activada la casilla "Ir A" se modificará la clase PanelPropiedades eliminando la siguiente sentencia:

```
if (tareas.getTareas().size() == 2) irA.setEnabled(false)
```

Esta sentencia desactivaba la casilla "Ir A" si la cantidad de tareas era igual a 2. Eliminándola se corrige el problema.

✓ También se ha debatido una idea sobre el modo en el que posiblemente se desarrollará los modos de presentación más adelante. La clase *VentanaPrototipoDialogo* que actualmente extiende la clase *JDialog* podría extender la clase *JPanel*. De este modo usando una sola ventana JDialog, habría que encargarse de cambiar el contenido de la misma modificando la posición y la visibilidad de los *JPanel*.

Pero como ya se ha dicho, es una idea que se tendrá en cuenta cuando llegue el momento de implementar los modos de presentación. De todos modos, es un avance importante de cara al futuro.

Planificación de actividades para esta iteración

- Seguir completando la memoria realizando todos los cambios necesarios para corregir los fallos y revisar las estimaciones que se han determinado hasta ahora.

- Corregir errores conocidos en la lógica de negocio del programa antecesor, en este caso referente a las propiedades de las tareas repetitivas.
- Continuar extendiendo la funcionalidad del prototipo de diálogo para permitir el funcionamiento de las tareas de orden "elección" y de cualquier tipo.

4ª Sprint Backlog: Del 30/01/2012 al 09/02/2012

En este cuarto *Sprint* se ha decidido mejorar los fallos más destacados en los botones para el control del mínimo y el máximo. También se corregirán otros fallos del panel de propiedades que surgen por culpa de una mala coordinación entre las tareas de elección y de tipo opcional. Debido a la proximidad de la siguiente reunión se ha dado preferencia al desarrollo de la memoria frente a la implementación de las tareas de orden indiferente.

Resultado de la evaluación

1. Se ha conseguido que las tareas de orden de elección funcionen correctamente en el prototipo de diálogo, pero se han encontrado algunos desperfectos a la hora de limitar al usuario a hacer cambios inadecuados en el panel de propiedades. Podemos ver un ejemplo en la siguiente imagen en el que se muestra que se permite que la propiedad "min" sea mayor que el "max" en las tareas de tipo repetitiva sin que salga ningún aviso. En otros casos, el mensaje informa correctamente de lo sucedido pero al aceptarlo, no realiza ningún cambio en las propiedades.

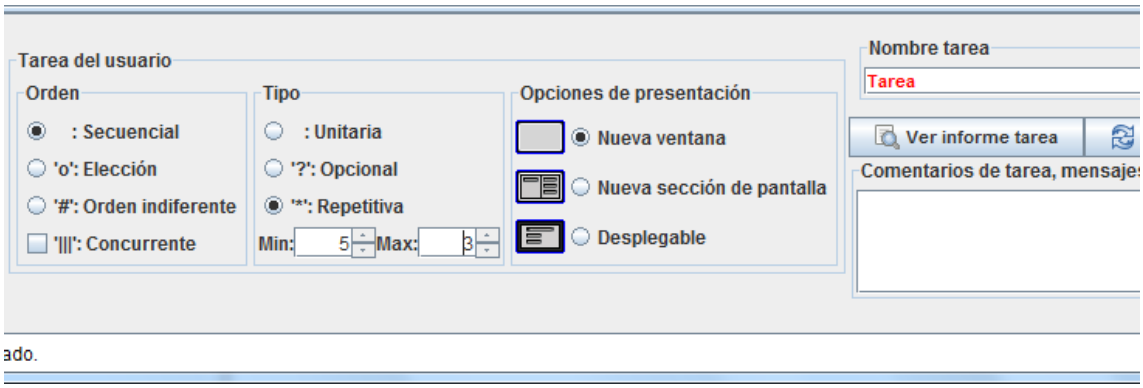


Ilustración 9: 4ª Sprint Backlog, resultado 1

El programa debería lanzar un mensaje informando de que la acción realizada no es permitida.

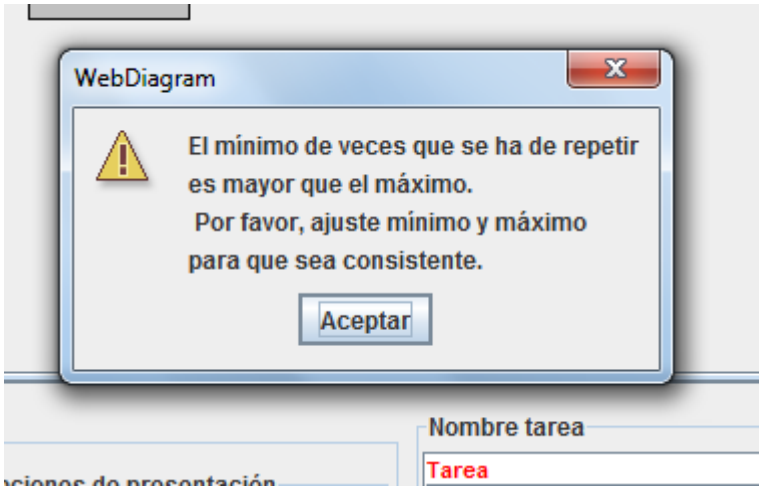


Ilustración 10: 4ª Sprint Backlog, mensaje max/min

Y a continuación el programa debería ajustar los valores automáticamente volviendo a los valores anteriores a la acción.

2. Ajeno al panel de propiedades, debe arreglarse un fallo que viene apareciendo desde la versión anterior de la herramienta, en el cual no sale el aviso del modelo de diálogo cuando solamente existe una tarea. Este mensaje avisa al usuario que debe estar en el modelo de diálogo para poder ejecutar el prototipo.

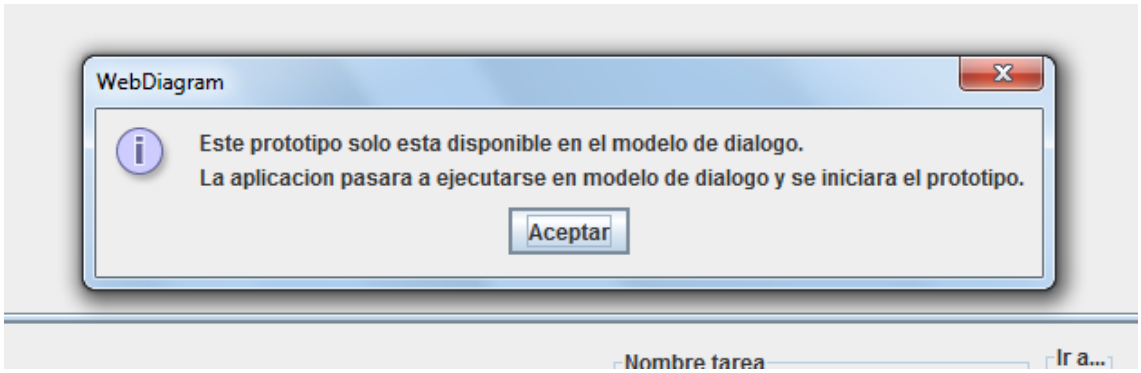


Ilustración 11: Mensaje del modelo de diálogo

Realmente el problema no es que no sale el mensaje, sino que nos da el aviso indicándonos la ausencia de subtareas y posteriormente sale el mensaje del modelo de diálogo.

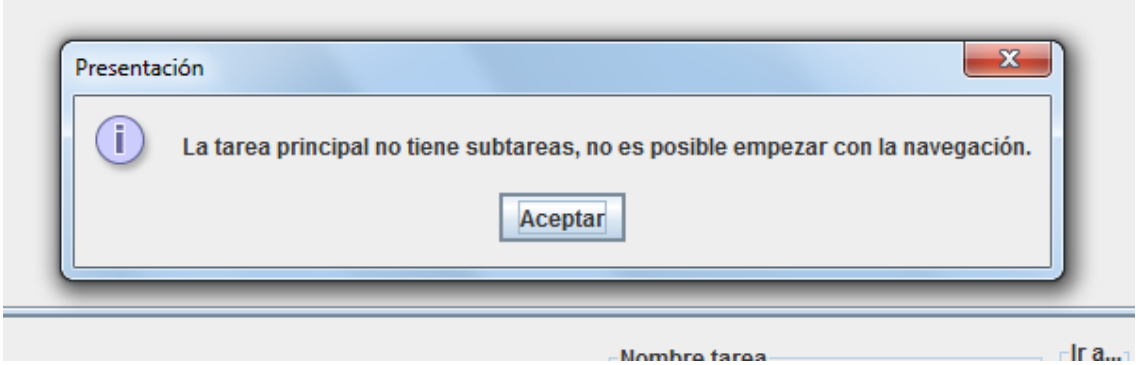


Ilustración 12: Mensaje de la ausencia de subtareas

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:45	0:00	0:45
		Realizar la reunión	11:00	1:30	9:30
		Recoger el acta	2:00	0:20	1:40
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		4:30	0:00	4:30
	Archivo	Mantenimiento	0:50	0:00	0:50
		Copias de seguridad	1:40	0:05	1:35
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		5:00	0:00	5:00
	Patrones Java		1:15	0:00	1:15
	Metodologías ágiles		2:30	0:00	2:30
Instalación	Instalación de las herramientas necesarias		0:15	0:00	0:15
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		1:30	0:00	1:30
Diseño	Diseño de la lógica de negocio		17:15	1:00	16:15
	Diseño del modelo de presentación		7:00	0:00	7:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		2:30	0:00	2:30
	Implementación de la lógica de negocio	Tareas de orden secuencial	1:00	0:30	0:00
		Tareas de elección	7:00	3:00	4:00
		Tareas de orden indiferente	10:00	0:00	10:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
		Tareas de orden concurrente	7:00	0:00	7:00
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		14:20	0:30	13:50
Documentación	Desarrollo de la memoria		49:15	0:50	48:25
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 5: 4ª Sprint Backlog

Decisiones de diseño

✓ Aunque el producto de esta iteración se haya realizado ignorando los problemas del panel de propiedades, se han tomado nuevas decisiones para evitar problemas en la ejecución del prototipo de diálogo. El programa permite que una tarea sea de orden de elección y de tipo opcional. Lo mismo ocurre con las tareas repetitivas, ya que ambos tienen por defecto un valor mínimo de 0. Una tarea de elección es automáticamente opcional ya que puede ser ejecutada o no, pero también hay que asegurar la ejecución de al menos una de las tareas de elección.

Teniendo en cuenta estas características, se ha decidido que se mantenga cierto control para que las tareas de elección no sean opcionales (que tengan un mínimo de 0). Para ello, se harán todos los cambios necesarios para que se muestren mensajes que informen sobre esta situación y se realizarán, en cada caso, cambios automáticos que facilitarán la interacción entre el usuario y la herramienta. Dichos cambios incluirán la modificación tanto del orden de la tarea como el mínimo de ejecuciones de la misma.

Decisiones de presentación

✓ Para corregir el **error número 2** se ha decidido alterar el orden de aparición de dichos mensajes, dando preferencia al aviso del modelo de diálogo. Solamente hay que cambiar de orden las sentencias “if” que comprueban estos casos.

Planificación de actividades para esta iteración

- Se corregirán los fallos de los botones Max/Min en las tareas repetitivas junto a otros ajustes que tienen que ver con el panel de propiedades, tales como no permitir que las tareas sean de elección y opcionales al mismo tiempo o repetitivas de mínimo 0.
- Además de corregir los fallos del panel de propiedades, también se corregirá el fallo del aviso del modelo de diálogo.
- A causa de la falta de tiempo en esta iteración, se ha decidido dedicar el siguiente Sprint a la corrección de fallos de menor grado y progresar en el

desarrollo de la memoria. También se comenzará a desarrollar el funcionamiento de las tareas de orden indiferente.

5ª Sprint Backlog: Del 10/02/2012 al 22/02/2012

En este *Sprint* se ha corregido el único fallo que se detectó en la ejecución de tareas de orden indiferente y se ha debatido una idea sobre la que se implementarán las tareas concurrentes. A la hora de corregir los errores del panel de propiedades, se ha tenido que reflexionar acerca de las posibles soluciones para poder optar por la más sencilla desde el punto de vista del programador, y la más manejable teniendo en cuenta las necesidades del usuario.

Resultado de la evaluación

1. Se ha obtenido una herramienta con fallos corregidos y además el funcionamiento de las tareas indiferente se ha implementado casi al completo, de no ser por un fallo que no se ha corregido por falta de tiempo. Este fallo conocido da lugar a una serie de anomalías en la ejecución de algunos modelos donde predominan las tareas opcionales.

Ilustración 13: Fallo de tareas indiferentes de tipo opcional

   En cualquiera de los 2 casos que se muestran en las imágenes, el prototipo finalizaría correctamente con solamente ejecutar la primera o segunda tarea, sin dar al usuario la oportunidad de ejecutar la otra tarea. Esto se debe a que al pulsar una tarea de orden indiferente siempre se comprueba si es correcto finalizar el prototipo. Si la comprobación es correcta el prototipo finaliza correctamente sin tener en cuenta que podría haber tareas opcionales que aún se podrían ejecutar.

2. Falta controlar el caso en el que el mínimo y el máximo tienen el valor 0. Sería más correcto limitar el valor del máximo para que no permita el valor de 0, ya que para el mínimo si es posible tener este valor.

3. También se ha detectado un detalle en el aviso que muestra cuando cambiamos las propiedades de una tarea para que sea de orden de elección y repetitiva.

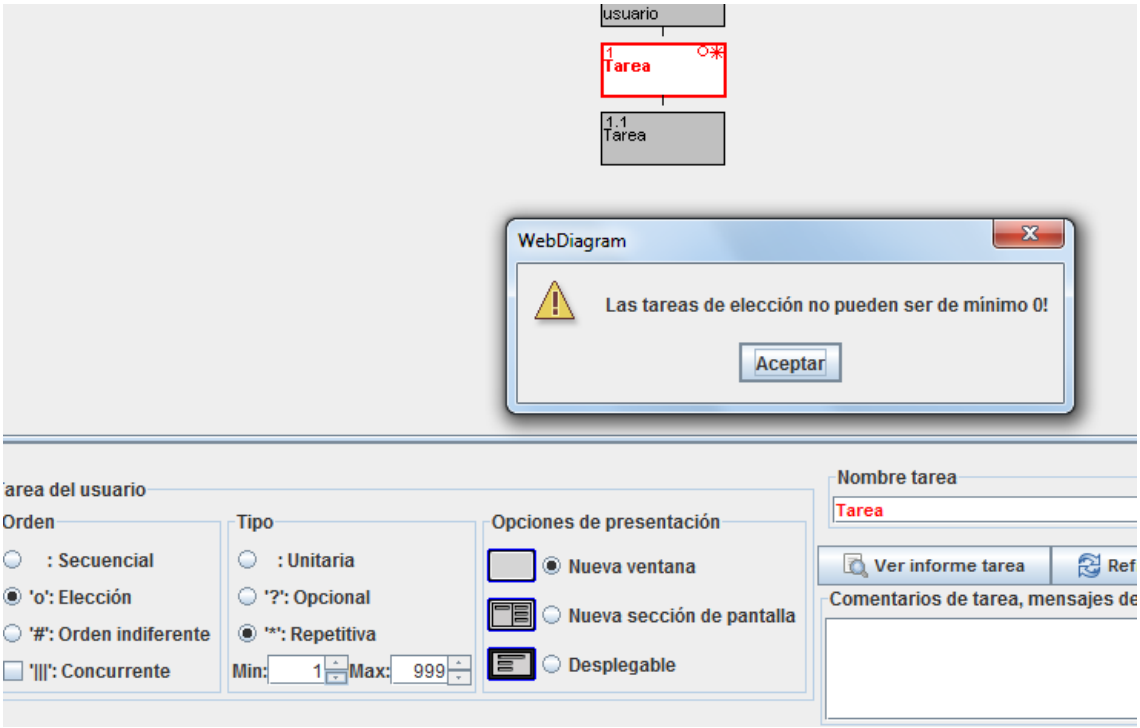


Ilustración 14: Fallo en las tareas de elección y de mínimo 0

Como se puede apreciar en la imagen, el mensaje muestra que una tarea no puede ser de elección y de mínimo 0, pero al mismo tiempo la casilla del mínimo muestra el valor de 1 en lugar de 0. Esto pasa porque se realiza el cambio del valor del mínimo antes de lanzar el mensaje.

4. Otro fallo de mayor grado se a encontrado cuando una tarea seleccionada no es opcional pero alguna de sus hermanas si, en cuyo caso no debería permitir el cambio al orden de elección.

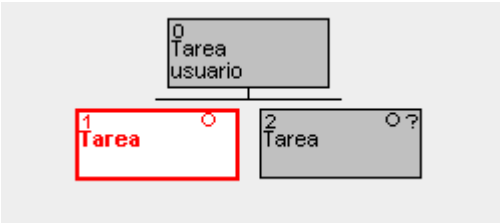


Ilustración 15: Fallo en tareas de elección y de tipo opcional

Actualmente éste es el resultado de dicho cambio y nos encontramos ante el caso que desde el principio se ha querido evitar.

5. Siendo el efecto similar al anterior problema, al seleccionar una tarea como elección y repetitiva, se cambia el mínimo a 1 pero no se aplican los mismos cambios necesarios a ninguna de las hermanas que sean repetitivas.

6. Se ha detectado un fallo heredado de la anterior versión del programa que permite la existencia tareas repetitivas cuando sus hijas son eliminadas. Debería dejarla en unitaria.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:45	0:05	0:40
		Realizar la reunión	9:30	2:00	7:30
		Recoger el acta	1:40	0:15	1:25
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		4:30	0:45	6:00
	Archivo	Mantenimiento	0:50	0:00	0:50
		Copias de seguridad	1:35	0:05	1:30
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		5:00	0:00	5:00
	Patrones Java		1:15	0:00	1:15
	Metodologías ágiles		2:30	0:00	2:30
Instalación	Instalación de las herramientas necesarias		0:15	0:00	0:15
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		1:30	0:00	1:30
Diseño	Diseño de la lógica de negocio		16:15	1:30	14:45
	Diseño del modelo de presentación		7:00	0:00	7:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		2:30	0:00	2:30
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	4:00	4:00	0:00
		Tareas de orden indiferente	10:00	0:30	2:30
		Tareas de orden concurrente	7:00	0:00	7:00
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		13:50	0:50	13:00
Documentación	Desarrollo de la memoria		48:25	1:30	46:55

	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 6: 5ª Sprint Backlog

Decisiones de presentación

✓ Se ha decidido hacer uso de distintos iconos en los mensajes que lanza la herramienta dependiendo de su carácter informativo.

Decisiones de implementación

✓ Para mejorar la gestión de los mensajes lanzados por WebDiagram, se ha tomado la decisión de agrupar todos los mensajes de error en un método que lanzará el aviso correspondiente al parámetro que recibirá. Gracias a este sistema será más fácil la traducción de dichos mensajes en un futuro.

La idea es que se enumeren los mensajes con un índice y hacer uso de la sentencia *switch* para mostrar los mensajes.

```
void lanzarMensaje ( int indice )
{
        switch (índice)
        {
                case 1:
                        JOptionPane.showMessageDialog…
                        break;
                case 2:
                …
        }
}
```

Planificación de actividades para esta iteración

- Se procederá a corregir todos los fallos detectados en esta serie de pruebas, que ya se han comentado anteriormente.
- Se realizarán cambios en los mensajes de alerta o de error para que muestren un icono apropiado dependiendo de la situación.
- Se terminará de completar el funcionamiento de las tareas de orden indiferente y se comenzará a desarrollar el de las tareas concurrentes. Para ello se realizarán todos los cambios necesarios tanto en el diseño de la interfaz como en la lógica de negocio.
- Las reuniones pasarán a ser los viernes a las 12:30 horas.
- Diseñar un método para lanzar mensajes de la herramienta con una sentencia *switch case*.
- Completar el código con comentarios que explican el funcionamiento de los arreglos.

6ª Sprint Backlog: Del 23/02/2012 al 08/03/2012

En este *Sprint* se ha implementado el control de los mensajes de error, para ello se han creado dos nuevas clases ya mencionadas anteriormente que interactuarán entre sí para lograr el objetivo.

También se ha corregido los fallos referentes a las tareas de orden indiferente y se ha implementado el funcionamiento de las tareas concurrentes. Para ello se ha modificado la interfaz del usuario para independizar el botón de tarea concurrente.

Resultado de la evaluación

1. Se ha detectado un fallo no visto anteriormente. Este fallo provoca la correcta finalización del prototipo cuando todavía quedan tareas repetitivas por ejecutar. Concretamente ha fallado al ejecutar el siguiente prototipo, donde la tarea repetitiva tiene un mínimo de ejecuciones de 2 y un máximo de 4.

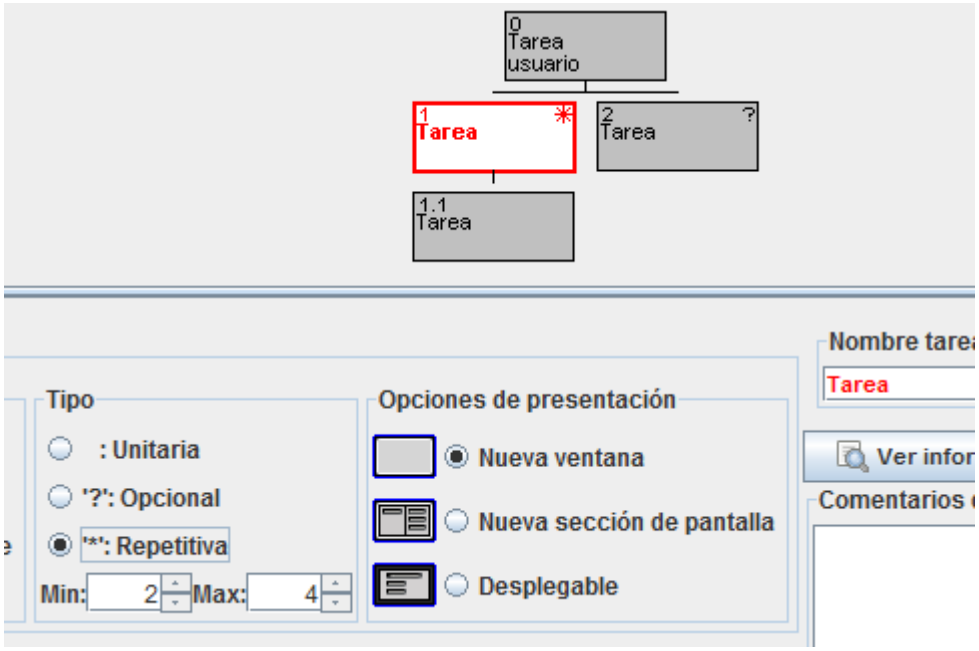


Ilustración 16: Fallo de tareas repetitivas

   Si se ejecuta el prototipo de diálogo, realizamos la primera tarea completamente y a continuación cerramos, el prototipo finaliza correctamente. Pero debería cancelar la ejecución ya que el mínimo de ejecuciones de la primera tarea es de 2.

2. El error que deja a una tarea padre ser repetitiva cuando sus hijas se han eliminado todavía no se ha corregido.

3. El tratamiento de errores no se ha implementado porque no se han fijado lo suficientemente bien todos los aspectos necesarios. Después de reflexionar sobre ello, se ha decidido que se realizará mediante una clase independiente, cuyo método público se encargará de lanzar el mensaje o confirmación adecuada. Todos los detalles están más abajo en la sección de decisiones de implementación.

4. Se ha detectado que cuando una tarea indiferente repetitiva es ejecutada, después de ejecutar otras tareas es posible volver a ejecutarla hasta llegar al máximo y esto no debería ser así.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:40	0:20	1:00
		Realizar la reunión	7:30	1:30	8:00
		Recoger el acta	1:25	0:15	1:10
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		6:00	0:30	5:30
	Archivo	Mantenimiento	0:50	0:00	0:50
		Copias de seguridad	1:30	0:15	1:15
Formación	WebDiagram		0:00	0:00	2:00
	Librerías gráficas		5:00	0:15	4:45
	Patrones Java		1:15	0:00	0:00
	Metodologías ágiles		2:30	0:00	2:30
Instalación	Instalación de las herramientas necesarias		0:15	0:15	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		1:30	0:30	1:30
Diseño	Diseño de la lógica de negocio		14:45	0:30	14:15
	Diseño del modelo de presentación		7:00	0:00	7:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		2:30	0:30	2:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:15
		Tareas de elección	0:00	1:10	0:00
		Tareas de orden indiferente	2:30	0:30	2:00
		Tareas de orden concurrente	7:00	0:00	7:00
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
Pruebas	Diseño de pruebas	10:00	0:00	10:00
	Ejecución de pruebas	13:00	0:30	12:30
Documentación	Desarrollo de la memoria	46:55	1:00	45:55
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 7: 6ª Sprint Backlog

Decisiones de diseño

- ✓ Hay que corregir el **fallo número 5** mencionado en la 5ª iteración, que cuando cambiamos el orden de una tarea repetitiva para que sea de elección, el valor mínimo pasa a ser 1 en la tarea seleccionada pero no realiza ningún cambio en las tareas hermanas. Se ha tomado una decisión importante al respecto y se ha establecido una regla general que afectará al cambio de propiedades de las tareas.

  Esta regla dicta que se deberán admitir los cambios de propiedad que afecten solamente a la tarea seleccionada lanzando un aviso informativo. En caso de que afecten a todas las hermanas, se deberá ignorar el cambio lanzando el previo aviso y así evitar cambios desastrosos no intencionales que podrían perjudicar el trabajo del usuario.

Decisiones de implementación

- ✓ Se ha querido mantener la programación orientada a objetos para facilitar a un futuro programador la modificación de los mensajes. La clase *Mensaje,* a pesar de su simplicidad, nos facilitará la creación de distintos tipos de mensaje gracias a sus atributos. Por otra parte, la clase *LanzadorMensajes* agrupará todos los mensajes y nos ofrecerá un único método que nos permitirá lanzar el mensaje deseado mediante un identificador de posición del vector representado por un entero. Cabe destacar que este método devuelve un valor entero que será útil cuando el mensaje lanzado sea una confirmación.

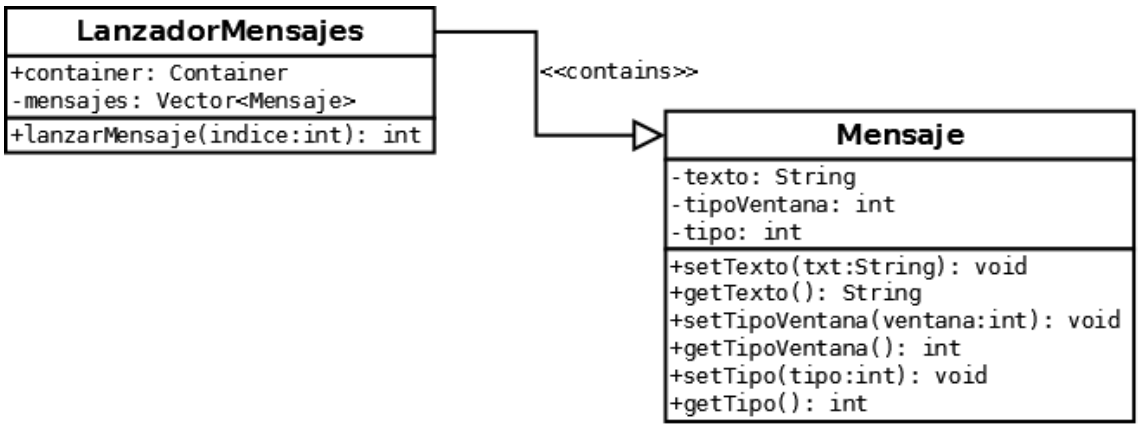


Ilustración 17: Diagrama de clases de LanzadorMensajes

Dependiendo del tipo de mensaje, se clasificarán en mensajes de error, mensajes de confirmación o pregunta, mensajes de aviso y mensajes de información. Para el dibujado

de los iconos se usarán las constantes ERROR_MESSAGE, QUESTION_MESSAGE, WARNING_MESSAGE e INFORMATION_MESSAGE respectivamente. Los atributos serán de tipo entero para mantener la compatibilidad con las funciones *showMessageDialog* y *showConfirmDialog* ofrecidas por Java que se usarán de manera implícita en el método *lanzarMensage*.

- ✓ Se ha tenido que tomar una decisión referente a la lógica de negocio de las tareas concurrentes. Se podría definir un nuevo atributo "booleano" para las tareas que indique la concurrencia o se podría mantener el atributo actual (indicado por el orden de la tarea) para mantener la compatibilidad con las anteriores versiones de la herramienta.

  Se han analizado los dos casos, pero este último es más apropiado ya que de lo contrario habría que modificar la implementación de la mayoría de las funciones de la herramienta. Así como la exportación a ficheros *XML*, el tratamiento de las tareas desglosadas, el simulador de tareas e incluso el prototipo de diálogo. Esto podría causar la incompatibilidad con las versiones anteriores de *WebDiagram*, con lo que se evitará implementar el primer caso si es posible.

Planificación de actividades para esta iteración

- Se corregirán los fallos anteriormente descritos y que no habían sido detectados hasta ahora.
- Se procederá a implementar las nuevas clases para el tratamiento de mensajes de error en el prototipo de diálogo.
- Se corregirá el fallo de las tareas indiferentes repetitivas para finalizar la implementación de las tareas indiferentes. Se prevé que éste error puede dar lugar a la aparición de errores similares, por lo que se le dará preferencia frente a las demás tareas.

7ª Sprint Backlog: Del 09/03/2012 al 15/03/2012

Después de dar el visto bueno al funcionamiento de las tareas de orden secuencial, de orden de elección y de orden indiferente, se ha pasado a implementar el modo de presentación en secciones.

Se ha mantenido al margen el desarrollo de las tareas concurrentes, ya que no se han fijado todas las características de este tipo de tareas. A pesar de ello, se han realizado los cambios necesarios para mejorar el aspecto del panel de propiedades y el botón de las tareas concurrentes.

Se ha corregido el movimiento del foco a través de los botones para facilitar al usuario la ejecución del prototipo mediante el teclado. Además, se ha actualizado el lienzo de la herramienta para que se seleccione automáticamente la tarea ejecutada en el prototipo.

Resultado de la evaluación

1. Observando detenidamente el foco de los botones durante la ejecución del prototipo de diálogo, se ha visto que no mantiene la posición correctamente. Como se puede ver en la siguiente imagen, después de ejecutar la primera tarea el foco debería pasar a la segunda tarea en lugar de permanecer en el primer botón.

Ilustración 18: Problema del foco

Corrigiendo esto y actualizando la posición del foco correctamente se dará al usuario una nueva forma de navegar por el prototipo de diálogo mediante el uso del teclado.

2. Se deben actualizar los botones de tipo cuando se cambian los valores del *spinner*. Por ejemplo, si el mínimo y máximo son 1 el tipo pasará a ser unitaria de forma automática.

3. Se ha detectado un extraño error que no se ha conseguido reproducir. A causa de este error, una tarea pasa a ser opcional solamente con seleccionarla.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	1:00	0:15	0:45
		Realizar la reunión	8:00	1:30	6:30
		Recoger el acta	1:10	0:10	1:00
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		5:30	5:00	5:00
	Archivo	Mantenimiento	0:50	0:05	0:45
		Copias de seguridad	1:15	0:05	1:10
Formación	WebDiagram		2:00	0:30	1:30
	Librerías gráficas		4:45	0:30	4:00
	Patrones Java		0:00	0:00	
	Metodologías ágiles		2:30	0:00	2:30
Instalación	Instalación de las herramientas necesarias		0:00	0:00	1:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		1:30	1:00	0:30
Diseño	Diseño de la lógica de negocio		14:15	0:30	13:45
	Diseño del modelo de presentación		7:00	0:00	7:00
Implementación	Definir la arquitectura		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Implementación de la interfaz		2:00	2:00	1:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:15	0:10	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	2:00	2:15	0:00
		Tareas de orden concurrente	7:00	2:45	4:15
		Presentación	9:00	0:00	9:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad "Ir A"	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		12:30	1:30	11:00
Documentación	Desarrollo de la memoria		45:55	2:00	43:55
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 8: 7ª Sprint Backlog

Decisiones de implementación

✓ Se ha aprobado la implementación de las nuevas clases que gestionarán el lanzamiento de los mensajes de aviso. Pero existen unos pocos mensajes que dependen del valor de ciertas variables y no han podido ser incluidas en la nueva clase *LanzadorMensajes* porque se requiere que el texto no sea variable. Por esta razón, se ha decidido dejar la resolución de este dilema para cuando se desarrolle el modo de presentación ya que ambos objetivos están ligeramente relacionados.

✓ Para implementar el nuevo modo de presentación en secciones se remplazará la clase *JDialog* por *JPanel*, de modo que la clase *VentanaPrototipoDialogo* heredará un *JPanel*. Por otro lado, la clase *GestorPorotitpoDialogo* heredará la clase *JDialog* y actualizará su contenido mediante paneles.

Planificación de actividades para esta iteración

- Se corregirán los fallos mencionados anteriormente, a excepción del extraño error que todavía se desconocen las causas que lo han podido provocar.
- Durante la navegación del prototipo se actualizará la selección de las tareas en el lienzo de *WebDiagram* para hacer un seguimiento de la ejecución.
- Se crearán nuevos *JPanel* en el panel de propiedades para dar un mejor aspecto al botón de concurrencia.

- Se empezará a desarrollar el siguiente modo de presentación que consiste en mostrar la ventana en secciones. Para ello se remplazará el uso de *JDialog* por un *ScrollPane* y se realizarán todos los cambios necesarios en la lógica de negocio.

8ª Sprint Backlog: Del 16/03/2012 al 22/03/2012

En esta iteración se ha dado un enfoque totalmente distinto a la implementación de los modos de presentación. A pesar de la suspensión de la batalla de pruebas de la reunión, se han tomado decisiones que han generado una reflexión sobre el camino optado para implementar los modos de presentación. Gracias a esta reflexión, en este *Sprint* se ha vuelto a implementar este aspecto mediante la creación de una nueva clase llamada *PanelPrototipoDialogo*.

Esta clase será realmente una copia de la antigua clase *VentanaPrototipoDialogo*, que ahora a pesar de mantener el nombre, cambiará totalmente su implementación. Además se han implementado las tareas concurrentes en el prototipo de dialogo a partir de las decisiones tomadas en la reunión.

Resultado de la evaluación

1. Se ha tenido que cancelar la batalla de pruebas sobre el nuevo modo de presentación por culpa de un error desconocido que congelaba toda la aplicación durante la ejecución del prototipo de diálogo.

2. Como el inconveniente ya mencionado ha perjudicado la batalla de pruebas de esta reunión, tampoco ha sido posible probar el efecto del foco sobre los botones.

3. Gracias a la independencia del panel de propiedades con el prototipo de diálogo, al menos se ha podido aprobar el nuevo aspecto del botón de la concurrencia.

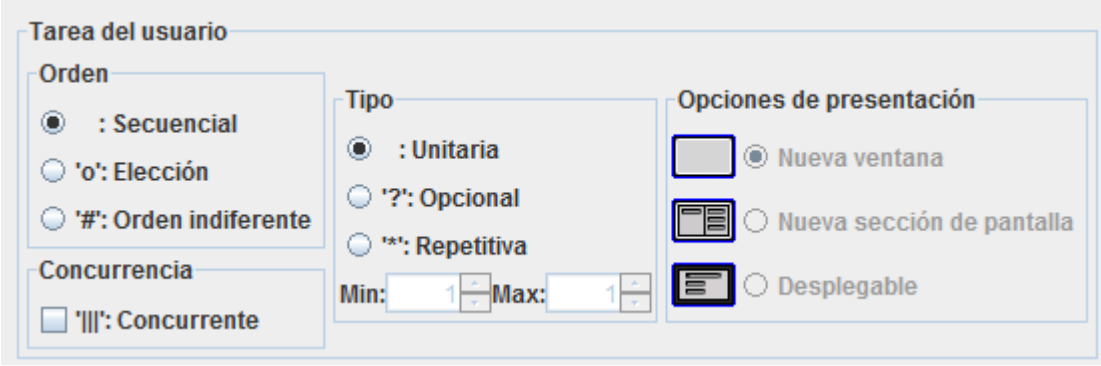


Ilustración 19: Aspecto del bóton de concurrencia

Para conseguir el efecto deseado se han incluido las secciones de orden y concurrencia en un nuevo *JPanel* con un *BoxLayout* que con un determinado atributo nos permite ordenar los elementos verticalmente.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:45	0:00	0:45
		Realizar la reunión	6:30	1:30	5:00
		Recoger el acta	1:10	0:25	0:45
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		5:00	0:30	4:30
	Archivo	Mantenimiento	0:45	0:05	0:40
		Copias de seguridad	1:10	0:05	1:05
Formación	WebDiagram		1:30	0:15	1:15
	Librerías gráficas		4:00	0:30	3:30
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:30	0:15	2:15
Instalación	Instalación de las herramientas necesarias		1:00	0:45	0:15
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:30	0:15	0:15
Diseño	Diseño de la lógica de negocio		13:45	0:30	13:15
	Diseño del modelo de presentación		7:00	1:30	5:30
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		1:00	0:55	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	4:15	0:10	4:05
		Presentación	9:00	3:30	5:30
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		11:00	1:00	10:00
Documentación	Desarrollo de la memoria		43:55	0:30	43:25

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 9: 8ª Sprint Backlog

Decisiones de diseño

✓ Se han tomado decisiones importantes en la detección de las causas que podrían dar problemas con las tareas concurrentes y los nuevos modos de presentación. Estas decisiones han sido tomadas principalmente por la propiedad que permite a las tareas concurrentes tener hijas.

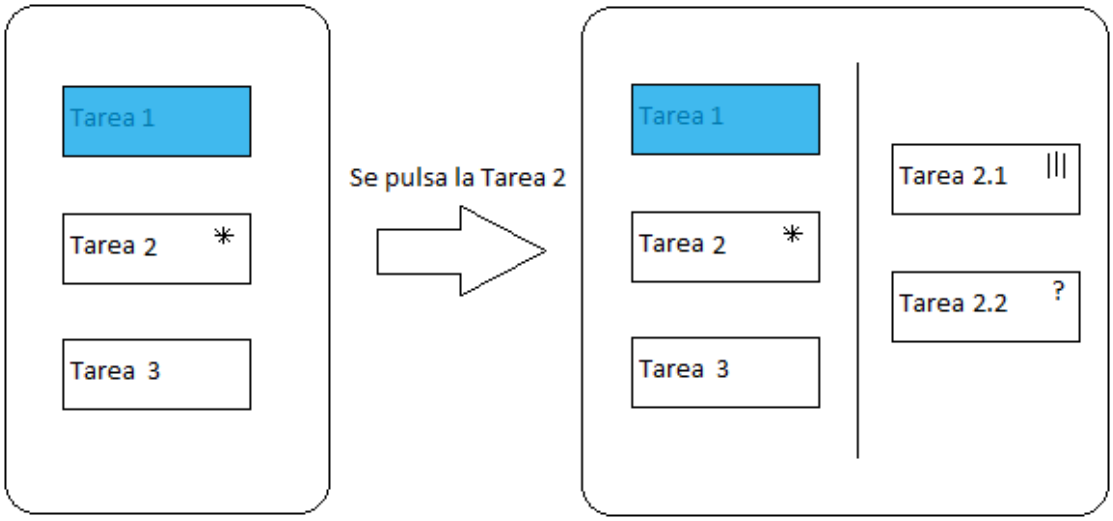


Ilustración 20: Prototipo del modo de presentación en secciones

Analizando la imagen anterior nos percatamos de que la 2. Tarea despliega sus hijas en una nueva sección, y además la 3. Tarea está activada porque la tarea anterior es una repetitiva con un mínimo de ejecuciones de 0. Estando en esta situación podríamos pasar a ejecutar la 3. Tarea o ejecutar la 2. Tarea.

Al ejecutar la 2. Tarea vemos que se mantiene activada en la sección anterior, ya que es una repetitiva y podemos volver a ejecutarla. En la nueva sección es opcional ejecutar cualquiera de las tareas, de modo que podríamos volver a ejecutar la 2. Tarea sin ningún problema. Al hacerlo tantas veces como se quiera, los botones de la nueva sección se reiniciarían sin notar cambio alguno en el prototipo hasta llegar al máximo de ejecuciones de la 2. Tarea. Dado su correcto funcionamiento, la lógica nos ha llevado ha dar el visto bueno a este comportamiento y calificar esta situación como un mal diseño por parte del usuario.

Planificación de actividades para esta iteración

- Se corregirá el fallo que ha evitado realizar las pruebas correctamente sobre el nuevo modo de presentación.

- Dado que se ha detallado más a fondo la interacción de las tareas concurrentes con los modos de presentación, para la próxima reunión se diseñara y se implementará una interfaz adecuada.
- Pensar en las posibles soluciones a los problemas que dan las tareas concurrentes con hermanas opcionales y padre de tipo repetitivo. De todos modos, la tarea descrita anteriormente arreglará o aportará soluciones fáciles a cuestiones de esta índole.

9ª Sprint Backlog: Del 23/03/2012 al 29/03/2012

En este Sprint se han corregido errores de menor grado detectados en la reunión. Se han buscado posibles soluciones mediante el uso de los *Layouts de Java* para crear el modo de presentación desplegable, pero la implementación se ha mantenido al margen para dar más tiempo a la decisión definitiva de las características de la concurrencia.

Resultado de la evaluación

Se ha visto que el trabajo realizado en esta iteración ha sido un gran avance para el proyecto. Sin embargo se han detectado errores que deben mejorarse.

1. Ha vuelto a aparecer el fallo detectado en anteriores iteraciones: Al ejecutar una tarea si las tareas que quedan por realizar son opcionales, el prototipo finaliza correctamente.

2. Al borrar una hija, el padre no debería tener activado el modo de presentación en el panel de propiedades.

3. En el *7. Sprint* se implementó la funcionalidad que permitía al usuario hacer un seguimiento de la ejecución del prototipo de diálogo mediante la selección automática de las tareas en el lienzo de la herramienta, a medida que se ejecutaban en el prototipo. Se ha detectado que al cambiar la selección de las tareas en el lienzo, no se actualiza correctamente el contenido del panel de propiedades.

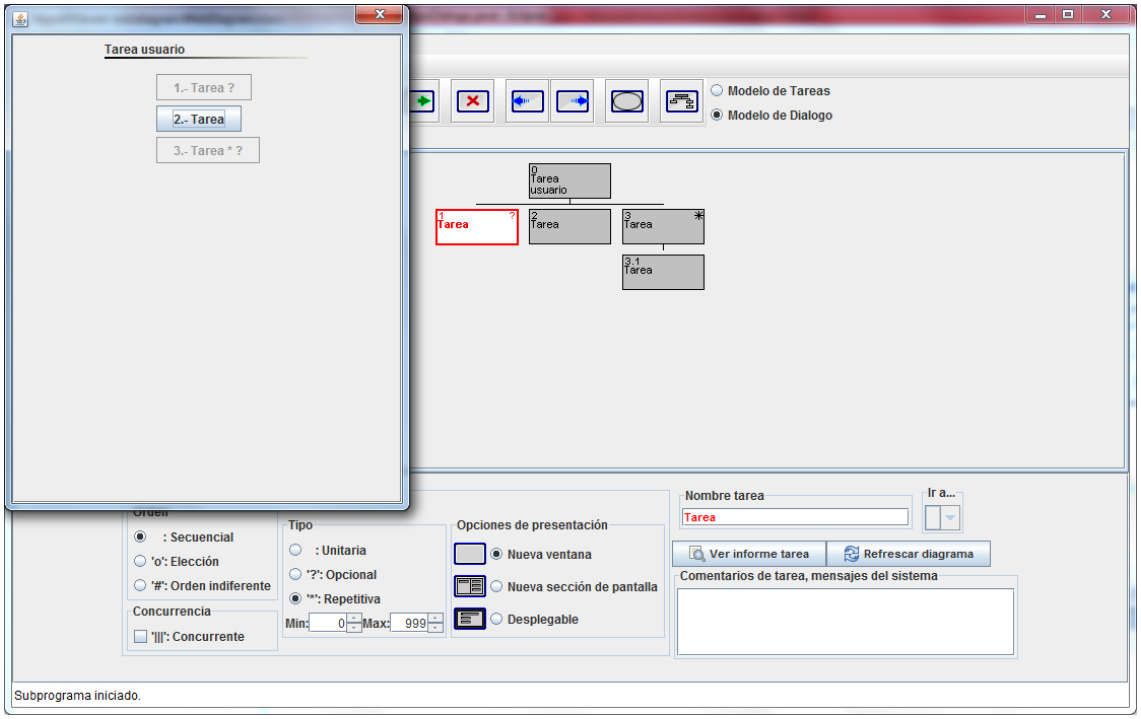


Ilustración 21: Funcionalidad de seguimiento del prototipo en el lienzo

Como se puede ver en la imagen, al ejecutar la 1. Tarea del prototipo, ésta se selecciona automáticamente en el lienzo pero el panel de propiedades no muestra la información adecuada. En este caso debería indicar que la tarea seleccionada es opcional.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:45	0:05	0:40
		Realizar la reunión	5:00	1:00	4:00
		Recoger el acta	0:45	0:05	0:40
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		4:30	0:45	3:45
	Archivo	Mantenimiento	0:40	0:00	0:40
		Copias de seguridad	1:05	0:05	1:00
Formación	WebDiagram		1:15	0:00	1:15
	Librerías gráficas		3:30	0:30	3:00
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:15	0:15	2:00
Instalación	Instalación de las herramientas necesarias		0:15	0:15	0:00
Captura de	Casos de uso		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
requisitos	Prototipo de la interfaz		0:15	0:00	0:00
Diseño	Diseño de la lógica de negocio		13:15	1:00	12:15
	Diseño del modelo de presentación		5:30	1:30	4:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	4:05	0:30	3:35
		Presentación	5:30	3:05	2:25
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		10:00	1:00	9:00
Documentación	Desarrollo de la memoria		43:25	0:30	42:55
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 10: 9ª Sprint Backlog

Decisiones de presentación

- ✓ Cuando el usuario está trabajando en una tarea desplegada en una nueva sección, las tareas de la sección padre deberán estar desactivadas, a excepción de las tareas concurrentes.

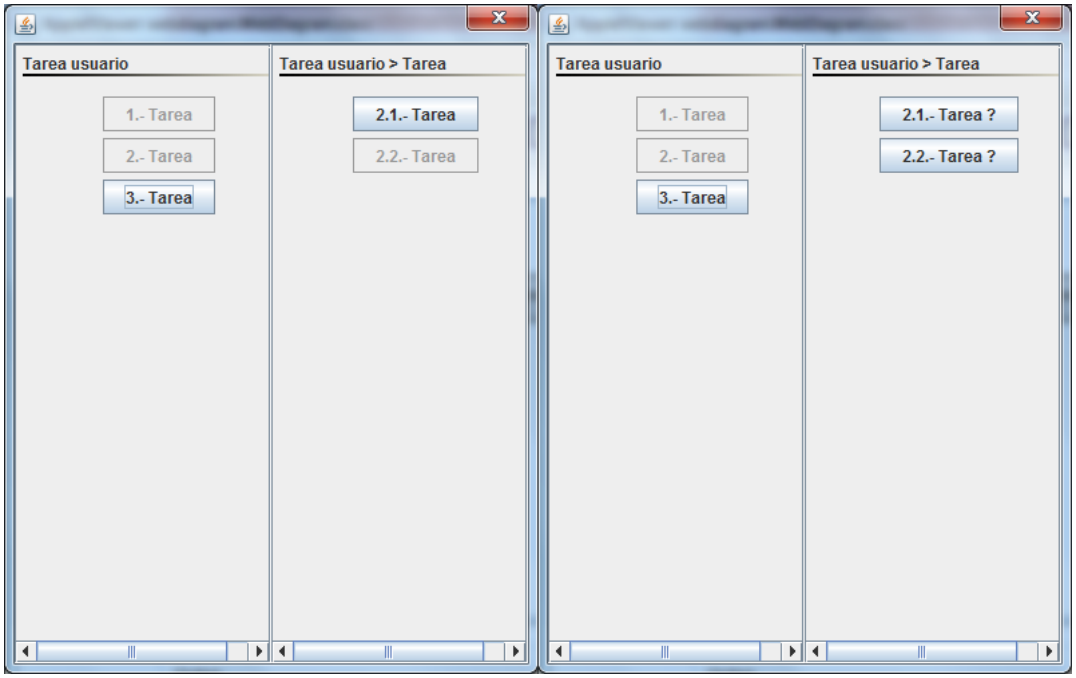


Ilustración 22: Activación de botones en secciones

La imagen de la izquierda muestra el caso e el que la 3. Tarea debería estar desactivada. Por otro lado, la imagen de la derecha muestra el caso en el que la 3. Tarea debería estar activada. Esto debe ser así porque las tareas de la nueva sección son opcionales y el usuario debería tener la posibilidad de no ejecutarlas y pasar a la siguiente tarea.

Decisiones de gestión

✓ Las estimaciones no se están haciendo específicas para cada iteración porque se está usando una variación de la metodología *Scrum*. Es posible cambiarlo pero se ha decidido seguir con el mismo procedimiento.

Planificación de actividades para esta iteración

- Corregir los errores detectados en la batalla de pruebas.
- A pesar de haber modificado el movimiento del foco a través de los botones, se dejó pendiente la implementación del mismo en el nuevo modo de presentación.
- Cambiar la implementación para que al crear una nueva hermana a una tarea concurrente, ésta sea de tipo unitaria.

10ª Sprint Backlog: Del 30/03/2012 al 03/04/2012

Este *Sprint* ha sido más corto que los demás dada la proximidad de la siguiente reunión. En él, se han hecho pequeñas mejoras en el prototipo de diálogo corrigiendo fallos en las tareas concurrentes y tareas de orden indiferente. Se ha analizado la dificultad de las tareas que se tendrán que completar en esta iteración y así crear un plan de desarrollo adecuado para Semana Santa.

Resultado de la evaluación

1. Todavía no se ha implementado la funcionalidad que deshabilita los botones de una tarea padre si una tarea hija es desplegada en una nueva sección. Pero las tareas concurrentes deberían estar habilitadas.

2. Se ha detectado un nuevo problema con las tareas indiferentes. Realizando una determinada secuencia de acciones sobre tareas de orden indiferente y repetitivas, se llega a una situación en la que todos los botones quedan desactivados sin llegar a finalizar el prototipo.

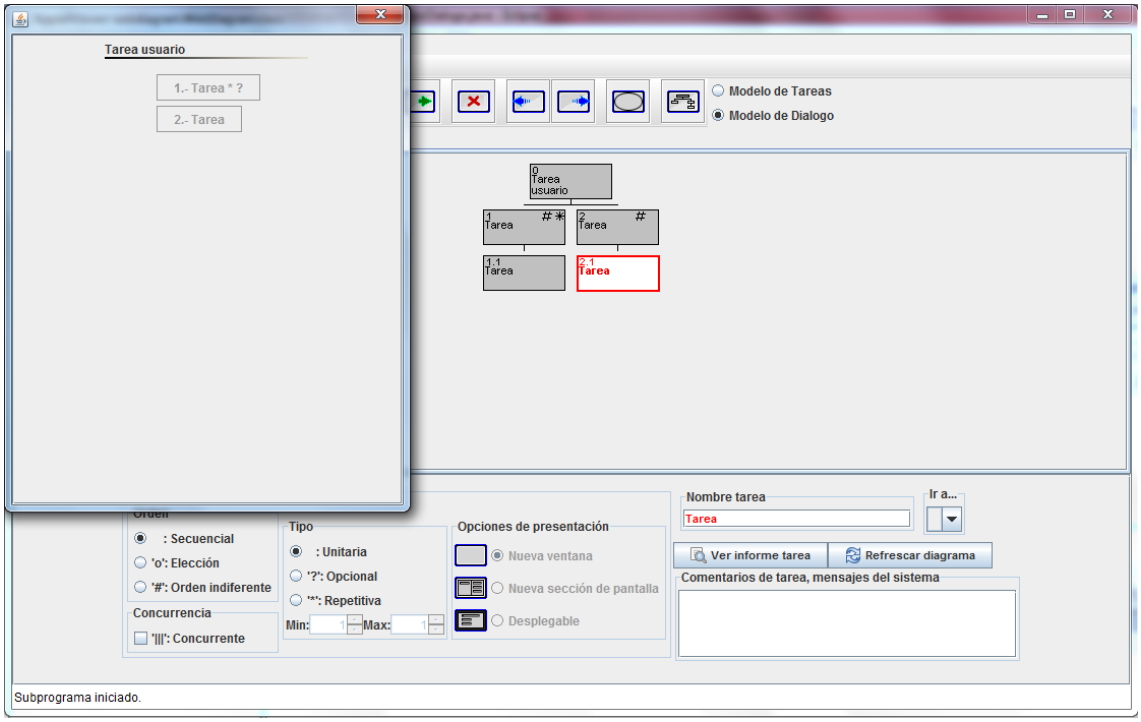


Ilustración 23: Todos los botones desactivados en el prototipo

En esta imagen se ve el estado en el que termina el prototipo si ejecutamos la siguiente secuencia de tareas: 1 - 1.1 – 2 - 2.1
Se ha comentado una posible y fácil solución para este problema que evitaría tener que realizar muchos cambios en el código. Solo habría que asegurar que todos los botones están desactivados al comprobar si todas las tareas han sido ejecutadas para finalizar correctamente el prototipo.

3. La ejecución de una tarea concurrente sin hijas debería lanzar una ventana informando sobre dicha acción.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:40	0:05	0:35
		Realizar la reunión	4:00	1:30	6:00
		Recoger el acta	0:40	0:10	0:30
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		3:45	0:30	3:15
	Archivo	Mantenimiento	0:40	0:05	0:35
		Copias de seguridad	1:00	0:00	1:00
Formación	WebDiagram		1:15	0:00	1:15
	Librerías gráficas		3:00	0:15	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		12:15	0:30	11:45
	Diseño del modelo de presentación		4:00	0:10	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:30
		Tareas de orden concurrente	3:35	0:00	3:35
		Presentación	2:25	1:45	1:30
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		9:00	0:15	8:45
Documentación	Desarrollo de la memoria		42:55	0:45	42:10

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 11: 10ª Sprint Backlog

Decisiones de diseño

- ✓ Se ha decidido que una tarea no puede ser finalizada si existen hijas concurrentes en ejecución. A su vez, se debe asegurar que el cierre de las tareas concurrentes se haga en orden, las hijas antes que el padre.

  Es decir, para que el prototipo de la siguiente imagen finalizara correctamente no valdría cerrar la segunda ventana, sino que antes habría que cerrar su hija concurrente (la tercera ventana).

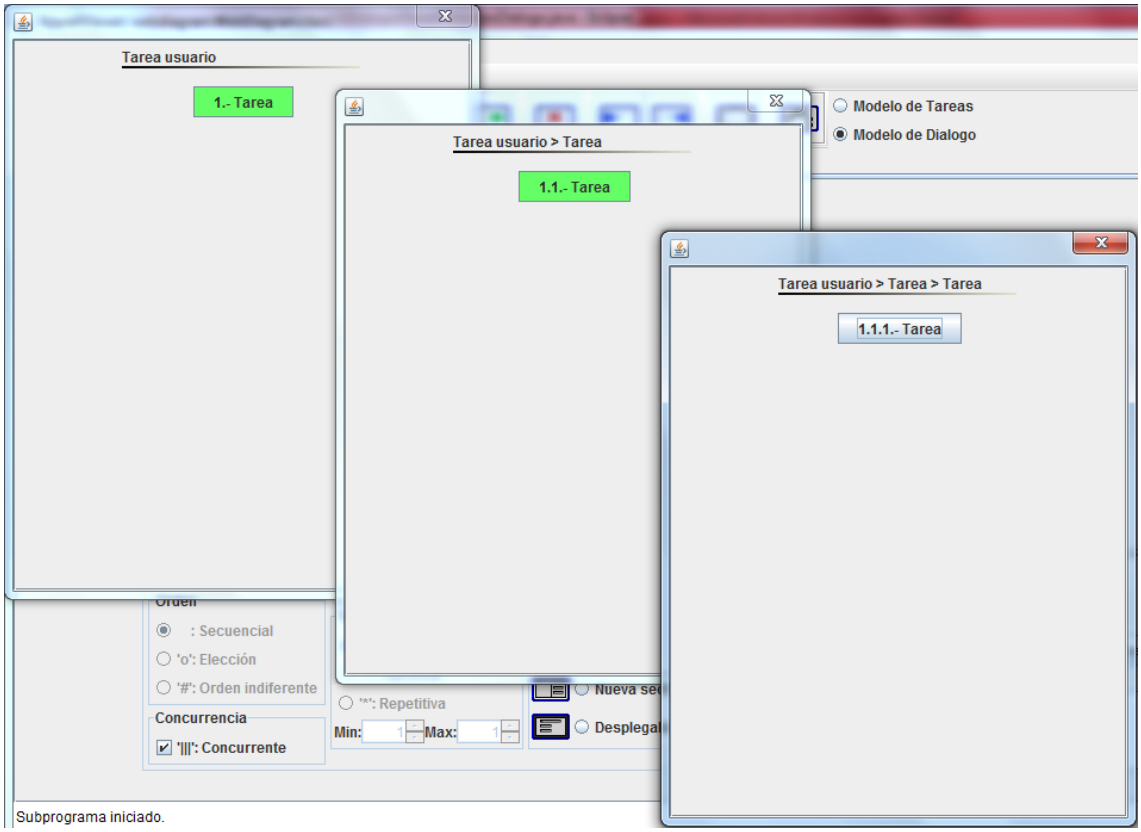


Ilustración 24: Orden de cierre de las concurrentes

Decisiones de presentación

- ✓ Se ha visto que los cambios realizados en el lienzo de la herramienta afectan a tiempo real en la lógica del prototipo, pero estos cambios no se reflejan gráficamente en los botones del prototipo. Se pretende aprovechar este comportamiento y arreglar la actualización de los botones.

  Finalmente se ha decidido ignorar este comportamiento, si se permiten los cambios en el diseño mientras se ejecuta el prototipo de dialogo se podría llegar a una situación desastrosa.

Planificación de actividades para esta iteración

- Se corregirán los fallos detectados en esta batalla de pruebas, a excepción del primer acuerdo descrito anteriormente y la implementación de las tareas concurrentes en una nueva sección.
- Se ajustará el tamaño de los paneles para que sea constante y para que la ventana se ajuste automáticamente gracias al uso de los *Layouts*.
- Se eliminará el máximo de secciones que se pueden crear, ya que se quiere evitar poner límites al diseño del usuario.
- Evitar que el foco automático se coloque sobre las tareas concurrentes, y se ejecuta una, el foco pasará a la siguiente tarea.
- Se cambiará la información referente al apartado "acerca de" de WebDiagram.

11ª Sprint Backlog: Del 04/04/2012 al 19/04/2012

En este *Sprint* se han corregido comportamientos inapropiados en las tareas concurrentes. Además se han simulado correctamente en nuevas secciones realizando un tratamiento adecuado en la visibilidad de los botones.

Se ha vuelto a implementar la funcionalidad que permite limitar la cantidad máxima de secciones, pero esta vez se calcula este valor dinámicamente mediante la división de la resolución de pantalla entre el tamaño de sección.

Resultado de la evaluación

1. Actualmente cuando se ejecuta una tarea concurrente sin hijas se lanza una ventanita para informar de su ejecución, pero al cerrar dicha ventana el prototipo de diálogo se esconde detrás de la aplicación principal de *WebDiagram*.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:35	0:05	0:30
		Realizar la reunión	6:00	1:30	4:30
		Recoger el acta	0:30	0:25	1:30
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		3:15	0:15	3:00
	Archivo	Mantenimiento	0:35	0:05	0:30
		Copias de seguridad	1:00	0:00	1:00
Formación	WebDiagram		1:15	0:00	1:15
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		11:45	0:10	11:35
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:05	0:30
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:15
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	3:35	1:20	2:15
		Presentación	1:30	0:20	3:00
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		8:45	0:15	8:30
Documentación	Desarrollo de la memoria		42:10	0:45	41:25
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 12: 11ª Sprint Backlog

Decisiones de diseño

- ✓ Las tareas concurrentes no son obligatorias, por eso se ha decidido que una ventana debe cerrarse manualmente si quedan concurrentes activas.

Decisiones de presentación

- ✓ Se ha decidido mantener el límite y el aviso de la cantidad máxima de secciones. Es más, se ha decidido calcular este valor mediante una división entre la resolución de la pantalla y el tamaño de sección. De esta manera, *WebDiagram* tendrá conocimiento de la cantidad de secciones que le será posible representar correctamente en el prototipo de diálogo. En caso de superar esa cantidad, la herramienta lanzará un aviso al usuario y el prototipo intentará ajustar las secciones para que se muestren de forma correcta mediante el uso de los *ScrollPane*.

Planificación de actividades para esta iteración

- Corregir el movimiento del foco en las tareas de elección repetitivas y al cerrar una sección.
- Implementar el control del orden de los cierres en las tareas concurrentes con hijas. Y en caso de que una ventana quede solamente con tareas concurrentes, esperar al cierre de la misma manualmente por parte del usuario.
- Tratar correctamente la visibilidad de botones en el modo de presentación de secciones.
- Implementación de las tareas concurrentes en nuevas secciones.
- Iniciar el diseño de la lógica de negocio de las funcionalidades "IR A" y las tareas del sistema.

12ª Sprint Backlog: Del 20/04/2012 al 25/04/2012

En este *Sprint* se han corregido los errores detectados en la batalla de pruebas. Al no haber realizado pruebas especificas sobre las tareas concurrentes y todas sus variantes, se ha aprovechado esta iteración para implementar la funcionalidad "Ir A" y las tareas del sistema.

Se han destinado pocas horas a la implementación de las tareas del sistema, dado que tienen la base de las tareas de elección. Por otro lado se han estimado más horas para la implementación de la funcionalidad "Ir A", básicamente porque todavía no se han fijado todas sus características y comportamientos. Por esta razón, solo se ha implementado la función básica que realiza un salto a la tarea indicada sin controlar el estado de las demás tareas, es una especie de prueba para tener una base sobre la que apoyar el resto del código.

Resultado de la evaluación

1. Se ha detectado un error en las tareas de elección cuando existe alguna hermana concurrente, como se ve en la siguiente imagen.

Ilustración 25: Error de tareas de elección y concurrentes

   Al realizar una de las tareas de elección, el prototipo no finaliza ya que existe una tarea concurrente que puede hacerse las veces que se quiera. El problema es que las tareas de elección no se deshabilitan y pueden volver a ejecutarse. **Se ha corregido el error en la reunión.**

2. En anteriores pruebas se pasó por alto un error en el panel de propiedades. Cuando deshabilitamos el *checkbox* de una concurrente, mantiene el orden anterior a pesar de tener hermanas de orden diferente.

Ilustración 26: Error al desactivar una concurrente

Experimentando con diferentes secuencias de acciones sobre el panel de propiedades, podemos llegar a la situación que se refleja en la imagen.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:30	0:00	0:30
		Realizar la reunión	4:30	1:30	3:00
		Recoger el acta	1:30	0:20	1:10
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		3:00	0:00	3:00
	Archivo	Mantenimiento	0:30	0:05	0:25
		Copias de seguridad	1:00	0:05	0:55
Formación	WebDiagram		1:15	0:00	1:15
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		11:35	0:30	11:05
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:15	0:05	0:10
		Tareas de orden indiferente	0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
		Tareas de orden concurrente	2:15	2:15	1:30
		Presentación	3:00	1:40	1:20
		Tareas del sistema	2:00	0:00	2:00
		Funcionalidad “Ir A”	5:50	0:00	5:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		8:30	1:00	7:30
Documentación	Desarrollo de la memoria		41:25	0:15	41:10
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 13: 12ª Sprint Backlog

Decisiones de implementación

- ✓ A pesar de no haber tenido tiempo para realizar pruebas, se ha deducido que puede haber errores con las tareas concurrentes en nuevas secciones debido al extraño comportamiento del prototipo en una prueba. Por eso se prestará especial atención en la implementación de la lógica de negocio de las tareas concurrentes.
- ✓ La implementación del modo de presentación desplegable se dejará para cuando se hayan arreglado los errores con las concurrentes y nuevas secciones. Este modo de presentación mantiene una relación directa con la presentación en secciones, por eso se quiere asegurar primero el correcto funcionamiento del mismo.

Planificación de actividades para esta iteración

- Se arreglarán los errores detectados en esta batalla de pruebas, y se revisará la implementación de tareas concurrentes.
- Se implementarán las funcionalidades de “IR A” y las tareas del sistema.

13ª Sprint Backlog: Del 26/04/2012 al 03/05/2012

En esta iteración se han arreglado los casos problemáticos detectados en la reunión. Se esperará a la especificación completa de la funcionalidad “Ir A”, que no se implementará hasta corregir todos los errores generados por las tareas concurrentes.

Se ha tratado de arreglar todos los errores al mismo tiempo, es decir, sabiendo que los errores provienen del mismo problema se han tomado medidas sobre ese problema para hacer desaparecer los errores derivados.

Resultado de la evaluación

1. Se ha detectado un error que deja activa la casilla de “elección” a pesar de lanzar el mensaje avisando que una tarea de mínimo 0 no puede ser de elección.

2. Se ha encontrado un error en el caso que se muestra en la siguiente imagen realizando esta secuencia: 2, 2.2, 1 y 1.1

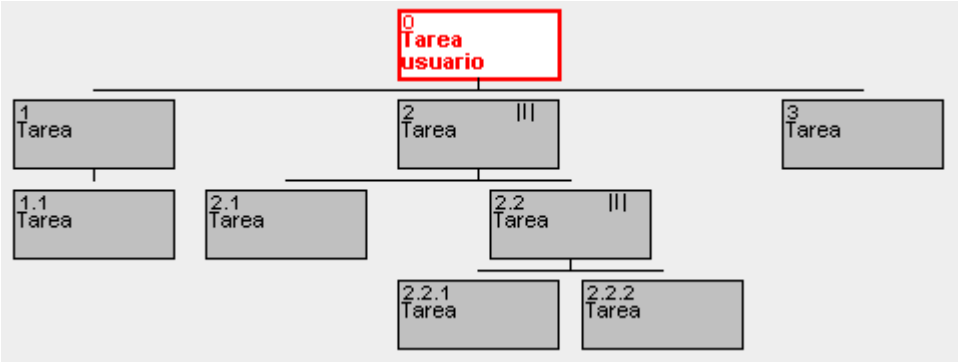


Ilustración 27: Fallo con concurrentes abiertas

Si ejecutamos el prototipo y realizamos dicha secuencia, nos damos cuenta que la ventana principal que representa la tarea raíz desaparece como si se hubiera cerrado. Cosa que no debería ocurrir.

3. En el siguiente caso el problema existe cuando ejecutamos una o más tareas concurrentes.

Ilustración 28: Diagrama de concurrentes con nueva sección

Que la 1. Tarea sea una concurrente en una nueva sección, nos permite llegar a una situación como la que podemos ver en la siguiente imagen.

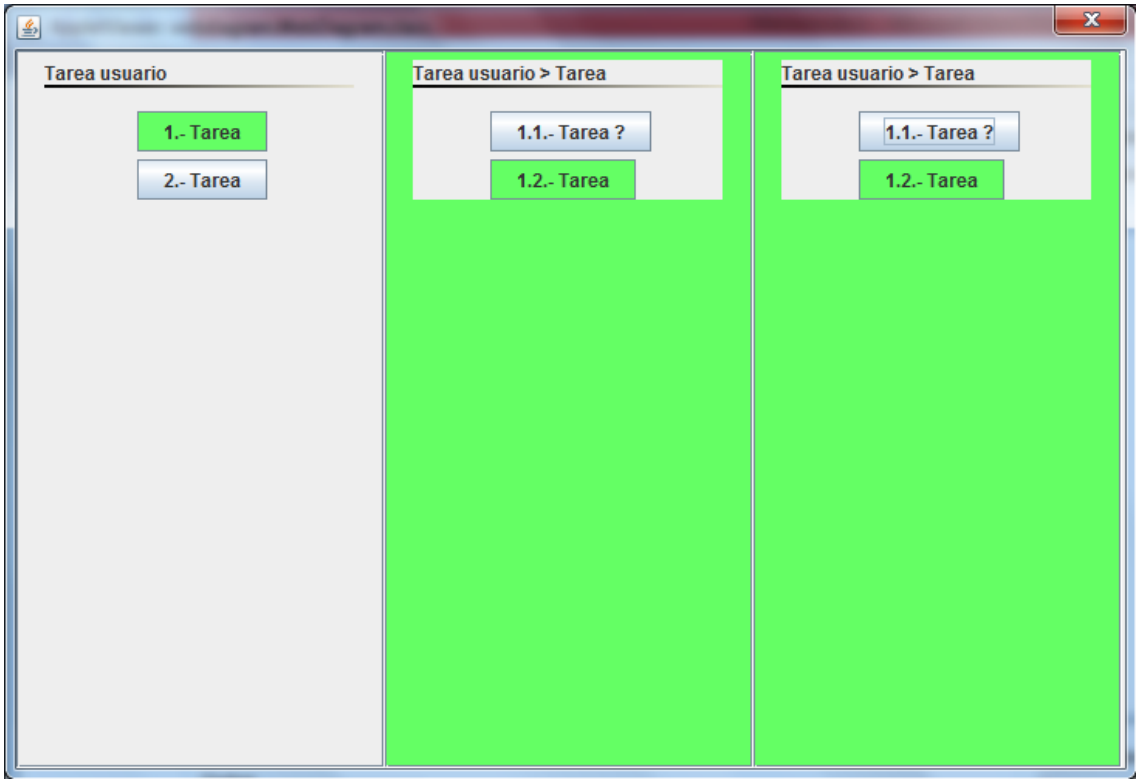


Ilustración 29: Error en las concurrentes en nueva seccion

Una vez en esta situación, debemos realizar la 2. Tarea obligatoriamente y las tareas opcionales pueden hacerse o no. Pero en cualquier caso si cerramos la ventana una vez realizada la 2. Tarea el prototipo se cancela, cuando realmente debería finalizar correctamente.

4. Por otro lado, el último caso de error pertenece a la categoría de las tareas repetitivas representado en la siguiente imagen.

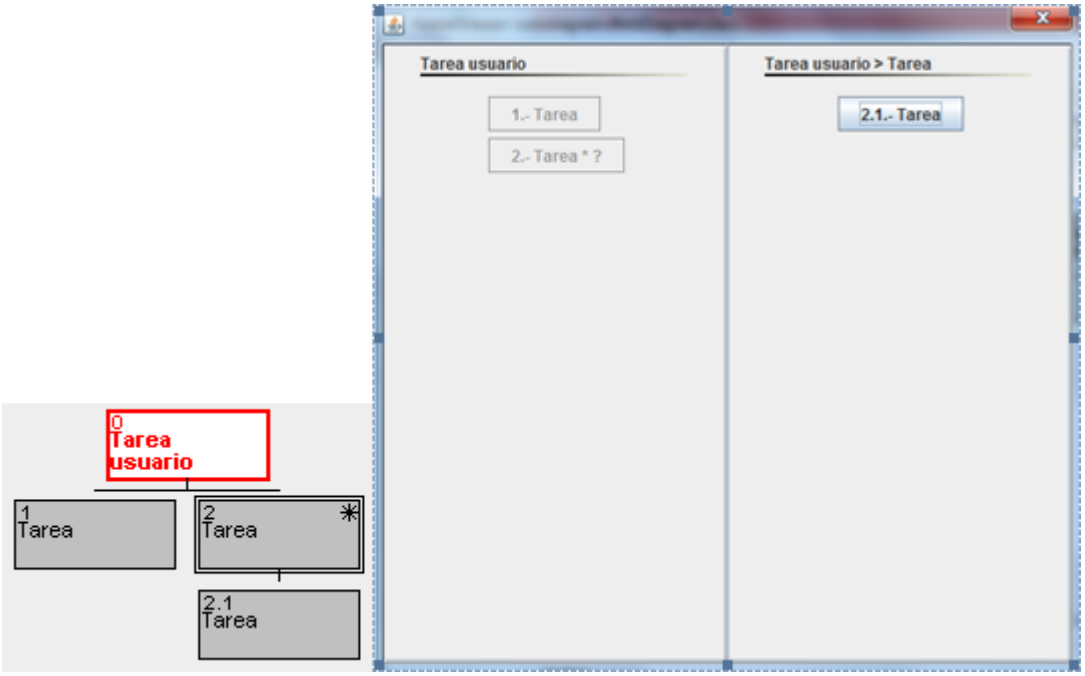


Ilustración 30: Error de tarea repetitiva en nueva sección

Nos encontraremos en esta situación concretamente cuando realicemos la 1. Y 2. Tarea. Si a continuación realizamos la tarea 2.1 vemos que el prototipo finaliza bruscamente, cuando lógicamente debería dar la opción de volver a ejecutar la 2. Tarea. Esto se debe a algunos cambios hechos en anteriores iteraciones.

5. Al activar una tarea del sistema, si la tarea anterior es de tipo opcional o repetitiva, el panel de propiedades se comporta de manera errónea. De modo que para evitar estos imprevistos, se cambiará el tipo de tarea automáticamente a unitaria.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:30	0:05	0:25
		Realizar la reunión	3:00	1:30	6:00
		Recoger el acta	1:10	0:10	1:00
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		3:00	0:30	2:30
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:55	0:00	0:55
Formación	WebDiagram		1:15	0:15	1:00
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		11:05	0:15	10:50
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:10	0:00	0:20
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	1:30	0:00	1:30
		Presentación	1:20	0:00	3:00
		Tareas del sistema	2:00	0:25	0:00
		Funcionalidad “Ir A”	5:50	0:30	5:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		7:30	0:30	7:00
Documentación	Desarrollo de la memoria		41:10	0:30	40:40
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 14: 13ª Sprint Backlog

Decisiones de diseño

✓ Se ha decidido permitir la ejecución del prototipo de diálogo sobre una tarea desglosada. La implementación que se ha llevado a cabo hasta este momento, da la posibilidad de hacer esto fácilmente ya que la ejecución siempre comienza desde una tarea (hasta ahora desde la tarea 0).

Planificación de actividades para esta iteración

- Se arreglarán todos los casos de prueba que han dado problemas. Aunque se sabe que más de uno proviene del mismo error, se analizaran todos los casos independientemente.
- Se analizará la integración de las tareas desglosadas con todo lo implementado hasta ahora para poder detectar incompatibilidades.

- Se hará una evaluación de dificultad sobre la nueva posible funcionalidad que permitirá ejecutar el prototipo de diálogo en una desglosada. Pero no se implementará todavía.

14ª Sprint Backlog: Del 04/05/2012 al 10/05/2012

En este *Sprint* se ha vuelto a implementar el prototipo de diálogo sobre una nueva estructura de datos. Se ha decidido que la estructura de datos más adecuada es el árbol, de modo que se ha eliminado todo lo relacionado con la estructura de pila en el prototipo.

Esto no quiere decir que no se ha aprovechado todo lo programado hasta ahora, se ha tratado de dejar la clase *PanelPrototipoDialogo* como estaba cambiando solo lo necesario para una correcta interacción con la nueva estructura.

La implementación de la funcionalidad "Ir A" se ha dejado para la siguiente iteración cuando se hayan aprobado todos los cambios sobre el prototipo.

Resultado de la evaluación

1. Se ha comprobado la estabilidad del prototipo de diálogo con las tareas no concurrentes, ya sea con el modo de presentación en ventanas o en secciones. Pero cuando las tareas concurrentes entran en juego, empiezan los problemas.

2. En el modo de presentación en secciones, el tratamiento de botones no se realiza correctamente y permite realizar tareas que no deberían hacerse hasta completar otras.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:25	0:05	0:20
		Realizar la reunión	6:00	2:00	4:30
		Recoger el acta	1:00	0:00	1:00
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		2:30	0:30	2:00
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:55	0:05	0:50
Formación	WebDiagram		1:00	0:00	1:00
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Diseño	Diseño de la lógica de negocio		10:50	0:30	10:20
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:20	0:15	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	1:30	1:30	5:00
		Presentación	3:00	0:10	2:50
		Tareas del sistema	0:00	0:05	0:00
		Funcionalidad “Ir A”	5:20	0:00	5:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		7:00	0:15	6:45
Documentación	Desarrollo de la memoria		40:40	0:40	40:00
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 15: 14ª Sprint Backlog

Decisiones de diseño

- ✓ Se ha terminado de especificar el comportamiento de la funcionalidad “Ir A”, pero se dejará su implementación para cuando se corrijan los problemas con las tareas concurrentes.

Decisiones de implementación

- ✓ Después de analizar todos los casos problemáticos que se han detectado en el área de las tareas concurrentes, se ha deducido que el problema es la ineficiencia del diseño de la lógica de negocio para tratar las tareas concurrentes. Básicamente el prototipo de diálogo maneja una pila de tareas representadas por ventanas. La ventana activa siempre es la que está encima de la pila. Cada vez que pasamos a una nueva tarea, incluimos una nueva ventana en la pila que pasará a tener la máxima prioridad. Cuando la terminamos, se elimina de la pila y se escoge la siguiente como ventana actual.

  Pero parece ser que con las tareas concurrentes no funciona de la misma manera, de hecho rompe la lógica de esta estructura. Gracias a las tareas concurrentes, podemos conseguir más de una ventana activa al mismo tiempo, situación difícilmente manejable para una pila cuya tarea actual siempre es el elemento superior.

Para arreglar este problema se ha pensado en **no incluir las tareas concurrentes en la pila**, pero esta acción traería otros nuevos problemas, y el propósito es dar con una solución completa. Por ejemplo, una tarea concurrente puede contener una hija no concurrente que en caso de realizarla se añadiría en la pila (ya que no es concurrente). De este modo tendríamos una hija de una concurrente como tarea actual, situación que no nos interesa, lo veremos en el siguiente caso:

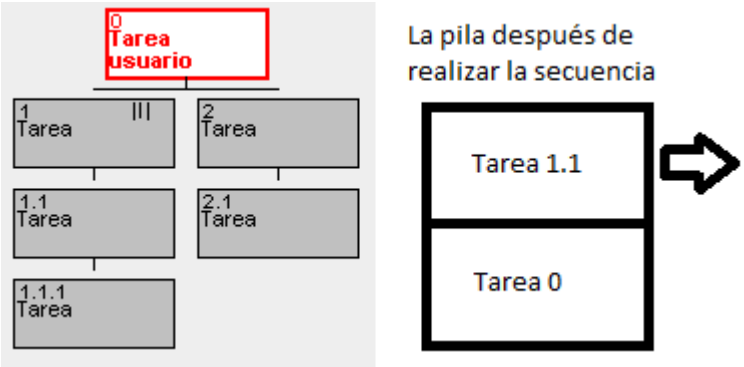


Ilustración 31: Estructura de pila

Ejecutamos la siguiente secuencia de tareas: 1, 1.1, 2, 2.1

Como resultado obtenemos una única ventana visible representando la tarea 1.1, cuando lo ideal sería que después de realizar la tarea 2.1, el prototipo nos llevara a la tarea 0. Pero en su lugar nos lleva a la tarea 1.1. Esto ocurre simplemente porque en la pila antes de la tarea 0, tenemos la tarea 1.1. El prototipo no se da cuenta de que realmente estas 2 tareas se están ejecutando de forma concurrente.

De modo que se ha pensado en una rápida solución**, no incluir en la pila las tareas hijas de las concurrentes**. Se ha descartado esta solución en el momento en que se ha pensado, ya que es indispensable mantener una referencia a todas las tareas.

Partiendo de la idea anterior, se les podría **atribuir una pila a cada una de las tareas concurrentes** donde se irían almacenando todas las hijas. Y yendo más allá, se podría definir el prototipo de diálogo como una tarea concurrente gigante, ya que cada prototipo es ejecutado en paralelo y totalmente independiente de los demás. Este concepto puede ayudar a generalizar la implementación de las tareas concurrentes y solucionar los problemas más graves.

Otra solución sería **descartar el uso de la pila** y moverse a través del prototipo mediante referencias de las tareas hijas hacia la tarea padre. **El tipo de estructura más adecuada en este caso sería un árbol, de hecho se ha optado por esta solución.**

Planificación de actividades para esta iteración

- Debido a la falta de tiempo, se hará una evaluación de posibles soluciones que permitan desarrollar correctamente el funcionamiento de las tareas concurrentes.
- Se dejará la implementación de la funcionalidad “Ir A” para la siguiente iteración.

15ª Sprint Backlog: Del 11/05/2012 al 17/05/2012

Esta 15ª iteración ha sido especialmente para implementar las primeras especificaciones fijadas acerca de la funcionalidad “Ir A”. Se ha implementado una función básica que permite al usuario desplazarse a una tarea padre por la que ya ha pasado el flujo de la ejecución.

Resultado de la evaluación

1. Se ha detectado un fallo en la interfaz del panel de propiedades a la hora de cambiar el mínimo de una tarea de elección. Si introducimos un valor mínimo de 0, el programa avisa correctamente comunicándonos que no es posible tener una tarea de elección de mínimo 0. El problema radica en que el nuevo valor mínimo reflejado por el panel de propiedades es 1, a pesar de que no se realiza el cambio en la lógica de negocio.

2. Las tareas de elección repetitivas solo las deja hacerlas una vez. Parece ser que el prototipo no espera a la decisión del usuario cuando quedan tareas repetitivas que pueden volver a ser ejecutadas.

3. En el modo de presentación de secciones, cuando se realiza una repetitiva debería cerrarse la anterior si ésta tiene hijas opcionales.

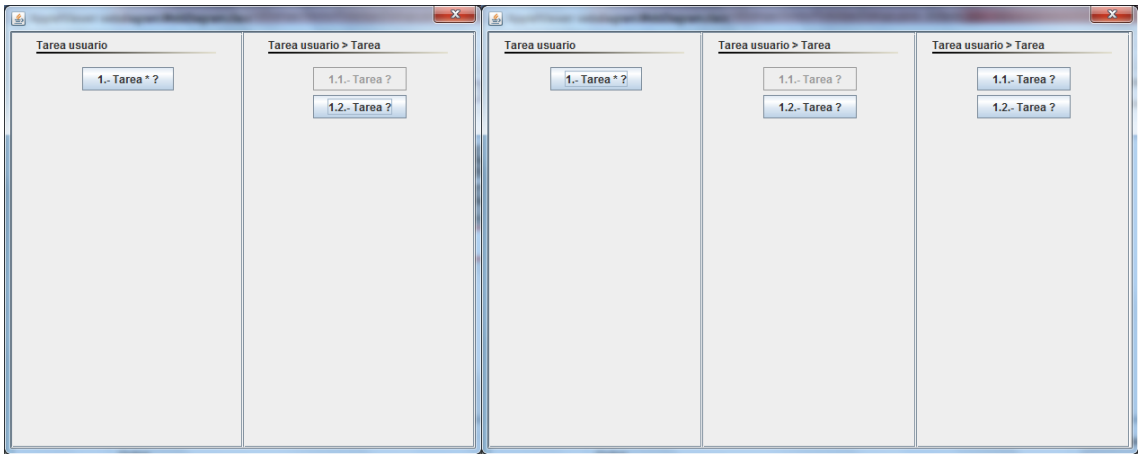


Ilustración 32: Cierre de secciones en tareas repetitivas

Nos encontramos ante la situación reflejada por la primera ventana, donde la 1. Tarea es repetitiva y ya se ha ejecutado una vez. Si la volviéramos a ejecutar, el prototipo debería cerrar la sección y crear una nueva, ya que las dos hijas son opcionales y es posible cerrarla. Pero el prototipo actúa creando una nueva sección pero manteniendo la anterior.

4. Cuando se vuelve a una ventana anterior y solamente quedan opcionales, debe esperar a la decisión del usuario en lugar de cerrar la ventana. Este fallo tiene cierto parecido con el error de las tareas de elección mencionado anteriormente.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:20	0:00	0:20
		Realizar la reunión	4:30	1:30	3:00
		Recoger el acta	1:00	0:10	0:50
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		2:00	0:15	1:45
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:50	0:05	0:45
Formación	WebDiagram		1:00	0:10	0:00
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:05
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		10:20	4:30	5:50
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:20
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	5:00	1:15	1:00
		Presentación	2:50	0:00	2:50
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	5:20	0:00	5:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		6:45	0:20	6:25
Documentación	Desarrollo de la memoria		40:00	0:20	39:40

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 16: 15ª Sprint Backlog

Planificación de actividades para esta iteración

- Se corregirán los fallos encontrados en la nueva implementación. Además se analizarán soluciones de estructuras de árbol aportadas por Juan Miguel, que posiblemente sean de utilidad en la implementación de la funcionalidad "Ir A".
- Se terminará de programar el tratamiento de los botones en las secciones y se cerrarán todas las ventanas cuando finalice el prototipo.

16ª Sprint Backlog: Del 18/05/2012 al 24/05/2012

En este *Sprint* se ha implementado a fondo una base para el funcionamiento de la funcionalidad "Ir A". Se ha implementado un método genérico para posteriormente evitar fallos en su comportamiento. Después de obtener un primer prototipo de esta funcionalidad, las pruebas guiarán el resto del desarrollo.

Además se ha arreglado el único fallo conocido y restante, el fallo de las tareas secuenciales detectado en la reunión.

Resultado de la evaluación

1. Las pruebas han sido satisfactorias en su totalidad de no ser por un pequeño fallo en las tareas secuenciales. El problema es que cuando las tareas concurrentes tienen hermanas secuenciales, se comportan como si también fueran secuenciales.

2. Se han hecho unos arreglos que no se especificaron en la reunión anterior, pero la funcionalidad "Ir A" todavía no está lista.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:20	0:00	0:20
		Realizar la reunión	3:00	2:00	8:00
		Recoger el acta	0:50	0:10	0:40
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:45	0:10	1:35
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:45	0:05	0:40
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		2:45	0:00	2:45

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Patrones Java		0:05	0:05	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		5:50	1:00	4:50
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:15
		Tareas de elección	0:20	0:20	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	1:00	1:55	0:00
		Presentación	2:50	1:10	1:40
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	5:20	1:00	4:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		6:25	0:10	6:15
Documentación	Desarrollo de la memoria		39:40	0:25	39:15
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 17: 16ª Sprint Backlog

Decisiones de diseño

✓ Se ha decidido que el salto que realiza el “Ir A” elimina por completo la ejecución actual. Si se salta a un punto por donde aún no ha pasado la ejecución del prototipo, se tendrá que reconstruir el árbol llamando al método que simula el clic de un botón.

Cuando las tareas concurrentes entran en juego la especificación puede variar, así que se dejará este tema para más adelante.

Planificación de actividades para esta iteración

- Se corregirá el fallo de las tareas secuenciales y se procederá a implementar a fondo la funcionalidad “Ir A”.

17ª Sprint Backlog: Del 25/05/2012 al 31/05/2012

A pesar de la falta de tiempo, en este *Sprint* se ha avanzado todo lo posible en la implementación de la funcionalidad "Ir A". Se han corregido los fallos detectados en la batalla de pruebas de la reunión. El objetivo se esta iteración ha sido conseguir una base funcional que se pueda analizar a fondo en la siguiente reunión.

Resultado de la evaluación

1. Existe un fallo cuando se cierran ventanas concurrentes, no tiene nada que ver con la funcionalidad "Ir A".

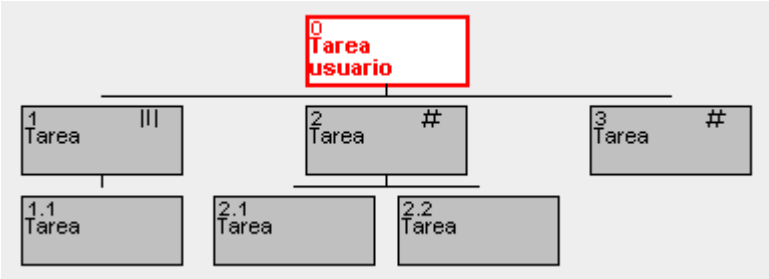


Ilustración 33: Fallo en el cierre de ventanas concurrentes

Realizando la secuencia; 1, 2, 1.1 nos damos cuenta que al cerrar la ventana concurrente nos vuelve a la tarea 0, cuando realmente deberíamos tener solamente la 2. Tarea a la vista.

2. Cuando realizamos una tarea con "Ir A" y eliminamos parte de la estructura de árbol, hay que tener en cuenta que también se deben eliminar los paneles creados en nuevas secciones. Más de lo mismo con las concurrentes.

   Cabe destacar que estos paneles no forman parte de la lógica una vez eliminados de la estructura. Y a pesar de que se pueda continuar con la ejecución, pueden acarrear serios problemas.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:20	0:05	0:15
		Realizar la reunión	8:00	2:00	6:00
		Recoger el acta	0:40	0:05	0:35
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:35	0:10	1:25
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:40	0:00	0:40
Formación	WebDiagram		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		4:50	1:00	3:50
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:15	0:30	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:20
		Presentación	1:40	0:15	1:25
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	4:20	3:35	3:00
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		6:15	0:10	6:05
Documentación	Desarrollo de la memoria		39:15	0:15	39:00
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 18: 17ª Sprint Backlog

Decisiones de presentación

✓ Se ha decidido que siempre que se cree una nueva ventana concurrente, aparezca más a la derecha para que se pueda apreciar la existencia de varias ventanas en ejecución.

Planificación de actividades para esta iteración

- Se corregirá los fallos encontrados y se avanzará en el desarrollo de la funcionalidad “Ir A”.
- Analizar la “marcha atrás” de la funcionalidad “Ir A”. Este término se refiere a cuando se da un salto a una tarea de una rama anterior.

18ª Sprint Backlog: Del 01/06/2012 al 07/06/2012

En esta iteración se ha continuado con la implementación de la funcionalidad "Ir A". No se han tenido en cuenta las tareas concurrentes, pero se ha hecho todo lo posible para que esta funcionalidad funcione bien con las tareas de cualquier orden.

Se ha dejado el debate de las tareas concurrentes para la siguiente reunión y será en el siguiente *Sprint* donde se implementarán las decisiones tomadas.

Resultado de la evaluación

1. El arreglo de un error de la anterior iteración ha generado un nuevo fallo a la hora de cerrar ventanas.

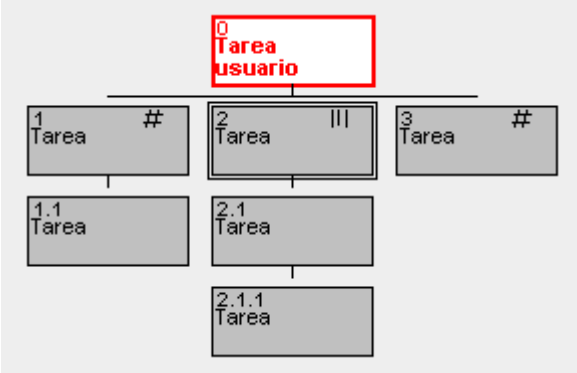


Ilustración 34: Error de visibilidad de ventana

Cuando realizamos la secuencia; 1, 1.1, 2, 2.1, 2.1.1 nos damos cuenta que al cerrarse la ventana actual, también perdemos de vista la ventana que representa la tarea 0.

2. Ha aparecido un error que no actualiza correctamente el estado de los botones cuando se realiza un salto con "Ir A". Se puede ver uno de los casos en la siguiente imagen.

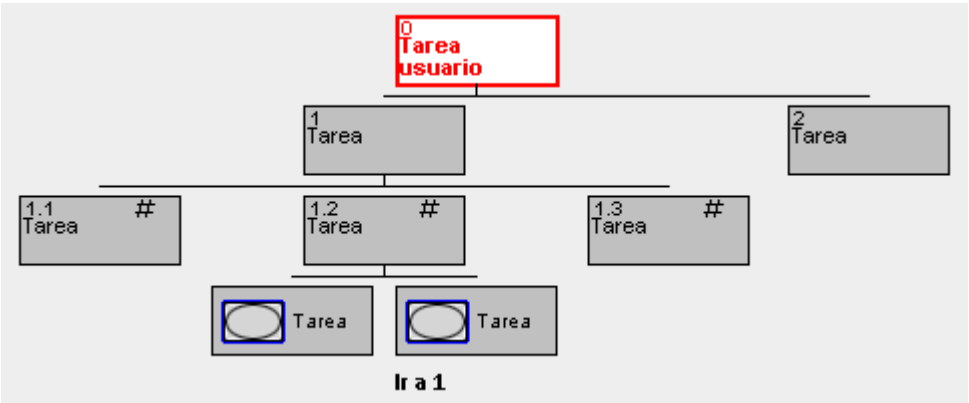


Ilustración 35: Error al salir de un Ir A

En este caso se puede realizar la 1. Tarea tantas veces como se quiera gracias al "Ir a 1" de una de las tareas. Si se ejecuta la otra tarea del sistema, la ejecución nos lleva a la tarea 0 pero con la 1. Tarea activa, en lugar de tener activa la 2. Tarea.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:15	0:05	0:10
		Realizar la reunión	6:00	1:30	4:30
		Recoger el acta	0:35	0:00	0:35
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:25	0:10	1:15
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:40	0:00	0:40
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		2:45	0:00	2:45
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:50	0:00	3:50
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:20	0:20	0:00
		Presentación	1:25	0:00	1:25
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	3:00	1:50	1:50
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		6:05	0:10	5:55
Documentación	Desarrollo de la memoria		39:00	0:20	38:40

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 19: 18ª Sprint Backlog

Decisiones de presentación

✓ Cuando se vaya a realizar un salto con "Ir A" deberá aparecer un mensaje indicando la tarea a la que saltará la ejecución del prototipo.

Planificación de actividades para esta iteración

- Se corregirán los fallos relacionados con la funcionalidad "Ir A", pero de momento no se entrará en la implementación de esta funcionalidad con las tareas concurrentes.

19ª Sprint Backlog: Del 08/06/2012 al 14/06/2012

En este *Sprint* se ha tratado de terminar de implementar completamente la funcionalidad "Ir A". Para ello se han corregido los fallos detectados en la reunión de esta iteración referentes a las tareas secuenciales y tareas indiferentes.

También se ha escogido y se ha desarrollado la mejor solución para mantener la interacción entre las tareas "Ir A" y las tareas concurrentes. El objetivo principal ha sido eliminar los errores más graves para poder dar esta tarea por finalizada. Los pequeños fallos se irán corrigiendo en las próximas iteraciones.

Resultado de la evaluación

1. Todavía está por limpiar las secciones y las ventanas concurrentes que quedan colgadas a causa de un salto de "Ir A". Estos paneles no forman parte de la lógica ya que se eliminan correctamente de la estructura de árbol, pero no se eliminan de la interfaz.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:10	0:05	0:05
		Realizar la reunión	4:30	1:30	3:00
		Recoger el acta	0:35	0:10	0:25
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:15	0:10	1:05
	Archivo	Mantenimiento	0:25	0:00	0:25
		Copias de seguridad	0:40	0:05	0:35
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		2:45	0:15	2:30

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:50	0:10	3:40
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	1:25	0:15	1:10
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	1:50	2:30	3:00
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		5:55	0:15	5:40
Documentación	Desarrollo de la memoria		38:40	0:20	38:20
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 20: 19ª Sprint Backlog

Decisiones de diseño

- ✓ Se han tomado decisiones acerca del “Ir A” sobre las tareas concurrentes. Decisiones que afectan a tareas “Ir A” que van a una concurrente tanto como a tareas “Ir A” que también son concurrentes.

  Cuando se está generando un árbol y existen ventanas concurrentes activas, no hay que cerrarlas ni eliminarlas de la lógica. Por ejemplo:

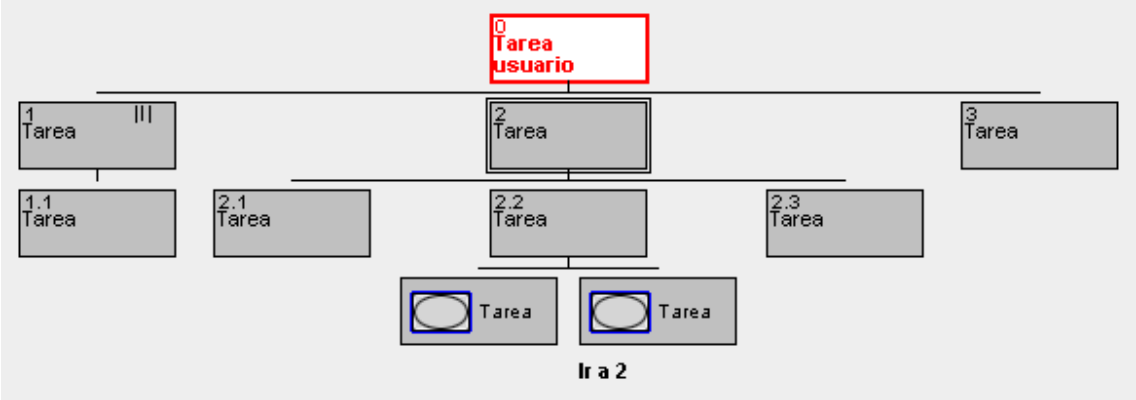


Ilustración 36: Ventanas concurrentes al hacer un Ir A

En este caso, al ejecutar la tarea "Ir A", si hubiera una ventana concurrente activa no debería eliminarse, ya que las tareas concurrentes pueden realizarse en paralelo.

- ✓ Otro caso interesante es cuando las tareas "Ir A" llevan a una tarea concurrente. En este caso aunque existan ventanas concurrentes activas, siempre se creará una nueva ventana y se mantendrá activa la ventana que representa al padre. Al ser concurrentes deberán permanecer visibles junto al padre, o en una sola ventana si las concurrentes se abren en una nueva sección.

- ✓ El último caso y el mas sencillo es cuando las tareas "Ir A" son al mismo tiempo tareas concurrentes. La solución consiste sencillamente en hacer que se comporte como si fuera una tarea secuencial a la hora de realizarla. La única diferencia está en que esta tarea estará siempre activa por el hecho de ser concurrente, y mostrará el mensaje informándonos de su ejecución.

Decisiones de implementación

- ✓ El método recursivo que se usa para realizar el salto y reconstruir el árbol trata a todas las tareas como secuenciales, hay que hacer una distinción de cada una para dar un trato específico para cada orden.

  La regla básica de la funcionalidad "Ir A" es dejar el prototipo en el mismo estado que se hubiera conseguido si hubieran ejecutado solamente las tareas necesarias para llegar al destino.
  De este modo, si se quiere saltar a una tarea secuencial, las tareas secuenciales anteriores deberían marcarse como realizadas. Si se quiere saltar a una tarea indiferente, las demás tareas indiferentes se dejarán intactas ya que no será necesario realizarlas para llegar a nuestro destino. Y en el caso de las tareas de elección todo funciona correctamente, se deshabilitarán todas a excepción de la tarea destino.

Planificación de actividades para esta iteración

- Se corregirán los fallos del "Ir A" para que funcione correctamente con las tareas secuenciales y de orden indiferente.
- Además, se implementará esta funcionalidad sobre las tareas concurrentes gracias a las especificaciones fijadas en la reunión.

20ª Sprint Backlog: Del 15/06/2012 al 21/06/2012

En esta iteración se ha tomado una decisión importante para corregir la mayoría de los errores que causan las tareas concurrentes con la funcionalidad "Ir A".

Además cabe destacar que se ha tenido que proceder a la recuperación de la información de todo el proyecto, ya que el ordenador de trabajo ha quedado inservible. No ha habido problemas ya que la gestión de las copias de seguridad se ha realizado durante todo el proyecto.

Resultado de la evaluación

1. Se ha encontrado un fallo con las concurrentes y el "Ir A". Cuando se quiere hacer un salto a una hija de una concurrente, el prototipo no crea una nueva ventana concurrente sino que se mete por la que ya está abierta.
2. Se han visto fallos con las concurrentes por no haber especificado todavía completamente el comportamiento con el "Ir A". Se tomará una decisión general para todos los casos de error.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:05	0:00	0:05
		Realizar la reunión	3:00	1:30	1:30
		Recoger el acta	0:25	0:05	0:20
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:05	0:00	1:05
	Archivo	Mantenimiento	0:25	0:05	0:20
		Copias de seguridad	0:35	0:05	1:00
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		2:30	0:00	2:30
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:40	0:20	3:20
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de	Tareas de orden	0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	negocio	secuencial			
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	1:10	0:00	1:10
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	3:00	1:40	1:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		5:40	0:20	5:20
Documentación	Desarrollo de la memoria		38:20	0:00	38:20
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 21: 20ª Sprint Backlog

Decisiones de diseño

- ✓ Se ha tomado una decisión y se ha fijado una regla general para las tareas concurrentes al hacer un “Ir A”. **Se mantendrán todas las tareas concurrentes excepto la concurrente origen de la que realizamos el salto “Ir A”, que desaparecerá completamente como si hubiese sido completada.**

Para ver un caso completo tenemos como referencia las siguientes imágenes.

Ilustración 37: Decisiones en tareas concurrentes y el Ir A

En un momento dado nos podemos encontrar con la situación de la imagen de la derecha. Dos concurrentes de la 1. Tarea abiertas una concurrente de la tarea 1.1 abierta. Si se pulsará la tarea del sistema que conlleva un “Ir A”, el prototipo debería realizar varias funciones teniendo en cuenta la regla general:

- En primer lugar debería crear una nueva concurrente por la que dirigir la ejecución.
- A continuación debería dejar intactas las demás concurrentes.
- Y finalmente debería cerrar completamente la concurrente origen de la que se ha ejecutado el "Ir A", en este caso las ventanas asociadas a la tarea 1 y la tarea 1.1. Así quedaría el prototipo:

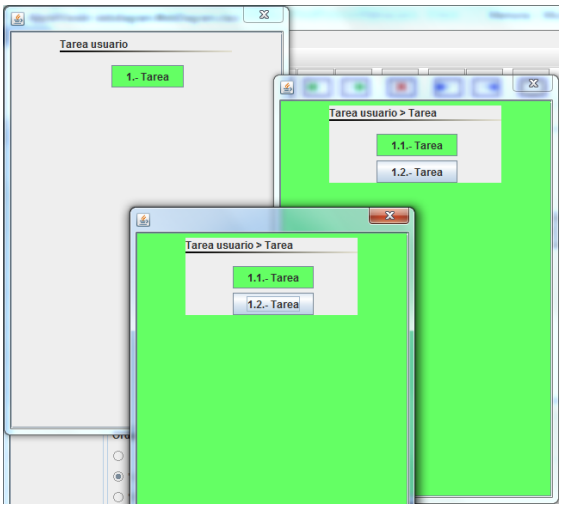


Ilustración 38: Resultado de las concurrentes al hacer el Ir A

Decisiones de implementación

✓ Para implementar la regla general descrita anteriormente, se ha decidido incluir un nuevo parámetro en la función recursiva que realiza el "Ir A". De este modo podemos detectar si nos encontramos ante una ventana origen para poder eliminarla del prototipo.

Decisiones de gestión

✓ El ordenador de trabajo ha quedado inservible, pero gracias a que en todas las iteraciones se haya llevado correctamente el plan de las copias de seguridad, se va a poder recuperar toda la información del proyecto que se habría podido perder.

Planificación de actividades para esta iteración

- Implementar la regla general especificada para las concurrentes al hacer un "Ir A".
- Desplazar la posición de las ventanas concurrentes a la hora de crearlas para poder visualizarlas mejor.

21ª Sprint Backlog: Del 22/06/2012 al 26/06/2012

En este *Sprint* se ha arreglado el fallo de no eliminar las concurrentes cuando se realiza un salto. La solución implementada en la anterior iteración solucionó la mayor parte de los errores. Además se ha arreglado la actualización de la numeración de los "Ir A" cuando se realizan cambios como mover o eliminar tareas.

Se han realizado unos cambios en el código para colorear las ventanas de las hijas de las concurrentes.

Resultado de la evaluación

1. Se ha encontrado un error en el siguiente caso al realizar el "Ir A" a la tarea 0. En la anterior iteración se arreglaron la mayoría de errores pero en este caso las ventanas concurrentes no se cierran al hacer el salto.

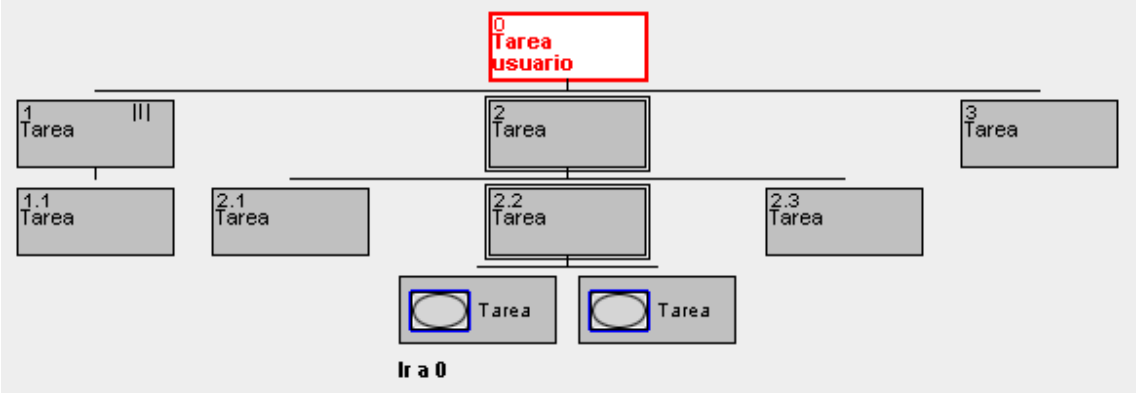


Ilustración 39: Caso de error en el cierre de ventanas al hacer el Ir A

2. La numeración de los "Ir A" no se actualiza correctamente cuando se mueven o eliminan tareas. En consecuencia podemos llegara situaciones como esta y tener serios problemas en el prototipo de diálogo.

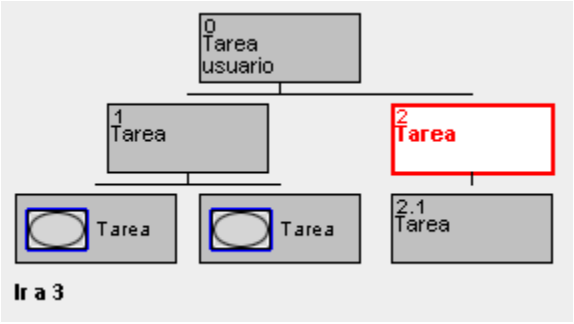


Ilustración 40: Fallo en la numeración Ir A

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:05	0:00	0:05
		Realizar la reunión	1:30	1:30	6:00
		Recoger el acta	0:20	0:00	0:20
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:05	0:00	1:05
	Archivo	Mantenimiento	0:20	0:00	0:20
		Copias de	1:00	0:50	0:20

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
		seguridad			
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		2:30	0:00	0:00
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:20	0:10	3:10
	Diseño del modelo de presentación		3:50	0:00	3:50
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	1:10	0:00	1:10
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	3:00	2:00	1:00
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		5:20	0:10	5:10
Documentación	Desarrollo de la memoria		38:20	0:00	38:20
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 22: 21ª Sprint Backlog

Decisiones de diseño

- ✓ Se ha decidido habilitar la posibilidad de ejecutar el prototipo de diálogo en una desglosada. Para ello solo hay que iniciar el prototipo dando como parámetro el identificador de la tarea que es desglosada.

  Al analizar esta funcionalidad se ha visto que podría haber problemas con los “Ir A”. Pero al final no habrá ningún problema ya que los “Ir A” están limitados para poder apuntar solamente a las tareas de la misma pestaña.

Decisiones de presentación

✓ Se ha decidido colorear de verde las ventanas que representan a las hijas de concurrentes. Pero para poder distinguirlas de las concurrentes y las demás ventanas, se ha decidido que el color se vaya aclarando cuanto mayor sea la profundidad de la tarea en el árbol.

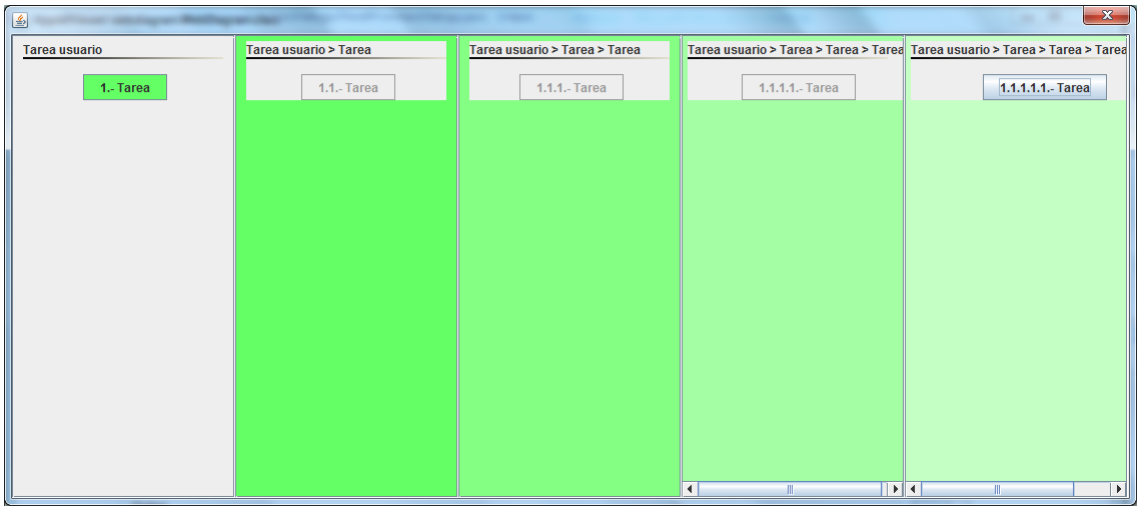


Ilustración 41: Secciones concurrentes coloreadas

Planificación de actividades para esta iteración

- Cambiar el código para colorear las ventanas de las hijas de las concurrentes.
- Hacer los cambios necesarios para que no haya problemas al ejecutar el prototipo de diálogo en una desglosada.
- Actualizar la numeración “Ir A” siempre que realicen cambios en la herramienta.

22ª Sprint Backlog: Del 27/06/2012 al 03/07/2012

En esta iteración se ha terminado de desarrollar la funcionalidad “Ir A” corrigiendo los errores que quedaban. También se ha arreglado la actualización de la numeración “Ir A”.

Con esto se puede decir que se ha completado el desarrollo del proyecto.

Resultado de la evaluación

1. Una sección queda colgada al hacer un “Ir A”. Solo forma parte de la interfaz, no de la lógica.
2. Se ha encontrado un error al hacer un “Ir A” sobre tareas de orden indiferente. No se ha conseguido reproducir el problema pero se tendrá en cuenta si salen nuevos errores relacionados.

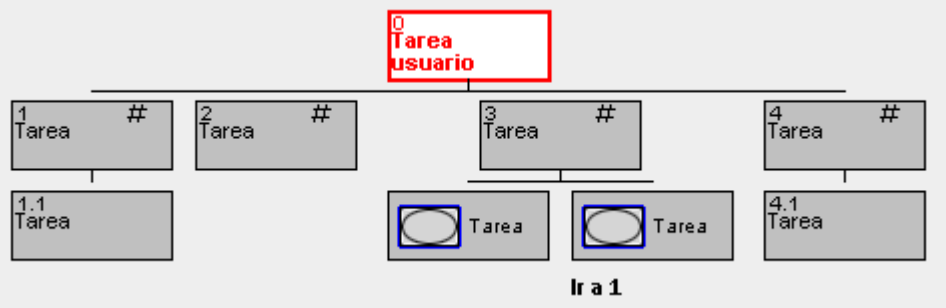


Ilustración 42: Error desconocido en el Ir A con las tareas indiferentes

3. No se ha arreglado completamente la actualización de la numeración “Ir A”.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:05	0:00	0:05
		Realizar la reunión	6:00	1:30	4:30
		Recoger el acta	0:20	0:05	0:15
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:05	0:00	1:05
	Archivo	Mantenimiento	0:20	0:00	0:20
		Copias de seguridad	0:20	0:05	0:15
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		0:00	0:00	0:00
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:10	0:00	3:10
	Diseño del modelo de presentación		3:50	0:10	3:40
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	1:10	0:00	1:10
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad "Ir A"	1:00	0:40	0:20
Pruebas	Diseño de pruebas		10:00	0:00	10:00
	Ejecución de pruebas		5:10	0:10	5:00
Documentación	Desarrollo de la memoria		38:20	0:00	38:20
	Creación del manual de usuario		2:00	0:00	2:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 23: 22ª Sprint Backlog

Decisiones de gestión

- ✓ Se ha decidido arreglar los fallos que quedan y dejar de lado el desarrollo para empezar a completar la memoria.

- ✓ Se ha decido cancelar el desarrollo del nuevo modo de presentación desplegable ya que no hay tiempo suficiente.

Planificación de actividades para esta iteración

- Arreglar los errores detectados en la batalla de pruebas para completar la funcionalidad "Ir A".
- Deshabilitar el modo de presentación desplegable en el panel de propiedades.

23ª Sprint Backlog: Del 04/07/2012 al 08/07/2012

En esta iteración se ha avanzado en la memoria y se han completado los *Sprint Backlogs*. Se ha dado el visto bueno al desarrollo del proyecto y se ha obtenido el producto completo, **WebDiagram 3.0**.

Resultado de la evaluación

1. Se ha completado el desarrollo del prototipo de diálogo y se ha conseguido un programa libre de errores gracias a todas las batallas de pruebas realizadas en cada iteración.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:05	0:00	0:00
		Realizar la reunión	4:30	1:30	3:00
		Recoger el acta	0:15	0:15	0:00
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:05	0:00	1:05
	Archivo	Mantenimiento	0:20	0:00	0:20
		Copias de seguridad	0:15	0:05	0:10
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		0:00	0:00	0:00
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	2:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		3:10	0:10	0:00
	Diseño del modelo de presentación		3:40	0:00	0:00
Implementación	Definir la arquitectura		0:00	0:00	0:00
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	1:10	0:15	0:00
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	0:20	0:45	0:00
Pruebas	Diseño de pruebas		10:00	0:00	0:00
	Ejecución de pruebas		5:00	0:15	0:00
Documentación	Desarrollo de la memoria		38:20	0:00	38:20

Fases	Tareas	Estimación anterior	Horas invertidas	Nueva estimación
	Creación del manual de usuario	2:00	0:00	2:00
	Elaboración de la presentación	15:00	0:00	15:00

Tabla 24: 23ª Sprint Backlog

Planificación de actividades para esta iteración

- Avanzar con la documentación del proyecto.

24ª Sprint Backlog: Del 09/07/2012 al 24/07/2012

Este último Sprint no ha formado parte de la metodología *Test-driven development*. Se ha definido como otra iteración más, pero realmente se ha destinado a la preparación de la presentación.

En la tabla de estimaciones se muestran las horas invertidas en la finalización de la memoria de la iteración anterior. Como la elaboración de la presentación va a quedar fuera de la memoria, se ha realizado una última estimación para este propósito.

Tabla de estimaciones

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
Gestión	Reuniones	Preparación del orden del día	0:00	0:00	0:00
		Realizar la reunión	3:00	1:30	1:30
		Recoger el acta	0:00	0:00	0:00
	Planificación inicial		0:00	0:00	0:00
	Realizar el Sprint Backlog		1:05	1:00	0:00
	Archivo	Mantenimiento	0:20	0:10	0:00
		Copias de seguridad	0:10	0:10	0:00
Formación	WebDiagram		0:00	0:00	0:00
	Librerías gráficas		0:00	0:00	0:00
	Patrones Java		0:00	0:00	0:00
	Metodologías ágiles		2:00	0:00	0:00
Instalación	Instalación de las herramientas necesarias		0:00	0:00	0:00
Captura de requisitos	Casos de uso		0:00	0:00	0:00
	Prototipo de la interfaz		0:00	0:00	0:00
Diseño	Diseño de la lógica de negocio		0:00	0:00	0:00
	Diseño del modelo de presentación		0:00	0:00	0:00
Implementación	Definir la arquitectura		0:00	0:00	0:00

Fases	Tareas		Estimación anterior	Horas invertidas	Nueva estimación
	Implementación de la interfaz		0:00	0:00	0:00
	Implementación de la lógica de negocio	Tareas de orden secuencial	0:00	0:00	0:00
		Tareas de elección	0:00	0:00	0:00
		Tareas de orden indiferente	0:00	0:00	0:00
		Tareas de orden concurrente	0:00	0:00	0:00
		Presentación	0:00	0:00	0:00
		Tareas del sistema	0:00	0:00	0:00
		Funcionalidad “Ir A”	0:00	0:00	0:00
Pruebas	Diseño de pruebas		0:00	0:00	0:00
	Ejecución de pruebas		0:00	0:00	0:00
Documentación	Desarrollo de la memoria		38:20	18:10	0:00
	Creación del manual de usuario		2:00	1:00	0:00
	Elaboración de la presentación		15:00	0:00	15:00

Tabla 25: 24ª Sprint Backlog

Planificación de actividades para esta iteración

- Completar lo que queda de la memoria.
- Elaborar y preparar la presentación.

4. La visión global

La metodología empleada produce resultados incrementales en la adquisición de requerimientos y en las decisiones y realización del producto obtenido. Esto se ha mostrado paso a paso tal y como se han producido en las iteraciones sucesivas del proceso. Sin embargo, para una mayor legibilidad y almacenamiento de la documentación, en este capítulo se van a mostrar en una visión global: los requerimientos, un ejemplo de un modelo de diagrama, y algunos modelos globales de la lógica de negocio.

4.1 Análisis de requisitos

En este capítulo se realiza el análisis de requisitos tanto funcionales como no-funcionales. Los requisitos funcionales son complejos por lo que se realiza una descripción exhaustiva de los mismos.

4.1.1 Requisitos funcionales

Teniendo en cuenta de que el objetivo del proyecto es implementar un prototipo de diálogo funcional, el único caso de uso que ha sido cambiado es el referente al prototipo de diálogo. A continuación se hace una descripción funcional de WebDiagram 3.0 y se detallan los requerimientos funcionales.

4.1.1.1 Descripción funcional de WebDiagram

La finalidad de la herramienta WebDiagram es facilitar la ejecución de algunas fases de la metodología InterMod. Está desarrollada para diseñar aplicaciones interactivas proponiendo un diseño centrado en el usuario para definir los requisitos, describir los diálogos persona-computador y evaluar los prototipos.

El **modelo de tareas de usuario** es la fase previa al *modelo de diálogo*. En él se establecen los pasos y orden en que el usuario deberá interactuar con el sistema con el fin de alcanzar los objetivos propuestos.

El modelo de tareas se representa en WebDiagram siguiendo una notación en árbol, donde los nodos de la jerarquía representan tareas del usuario y se caracterizan por:

- una numeración ordenada con respecto a los padres, hijos y hermanos.
- un nombre descriptivo sobre dicha tarea.
- unos símbolos que permiten especificar:
  - ✓ El orden de ejecución de la tarea con respecto a sus hermanas.
  - ✓ El tipo de la tarea.

Ilustración 43: Ejemplo de una tarea del usuario

La especificación del orden y tipo de las tareas está descrita mediante símbolos que aparecen en la esquina superior derecha del nodo que representa la tarea, como se muestra en la ilustración 52. A continuación se nombran las diferentes opciones de orden y tipo:

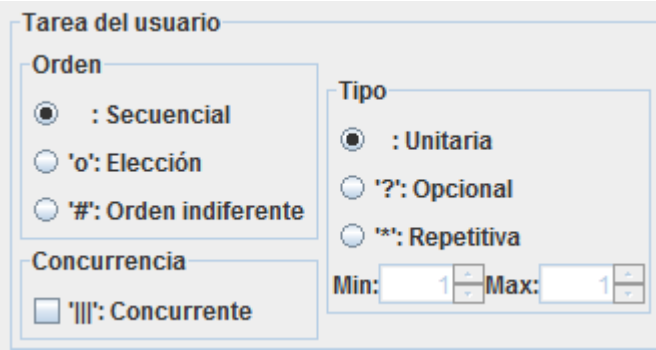


Ilustración 44: Orden y tipo de tareas

- Orden entre las tareas:
  - ✓ Secuencial (sin símbolo)
  - ✓ Elección (o)
  - ✓ Indiferente (#)
- Tipos de tareas de usuario:
  - ✓ Unitaria (sin símbolo)
  - ✓ Opcional (?)
  - ✓ Repetitiva (*)

A estas características hay que añadir una más que facilita el diseño de aplicaciones de mayor tamaño: desglose de tareas. Esta opción permite marcar una tarea como desglosada, de tal manera que se pueda seguir el diseño en una pestaña diferente, tomando dicha tarea como la raíz desde la que seguir creando el modelo.

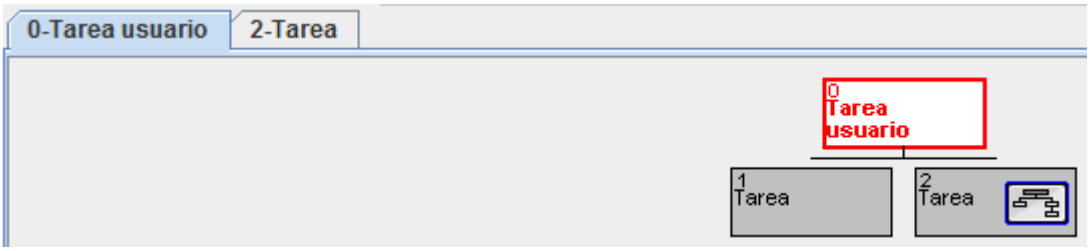


Ilustración 45: Ejemplo tarea desglosada

El ***modelo de diálogo*** incorpora las actuaciones o comunicación del sistema con el usuario y se indican los desvíos de la navegación prevista. Además se establece una forma de presentación sencilla (en ventanas nuevas o en nuevas secciones) para cada tarea. El modelo de diálogo será un grafo creado a partir del modelo de tareas ya creado.

En el modelo de diálogo debemos diferenciar entre las *tareas de usuario* y las *tareas de sistema*. Las tareas de usuario están numeradas mientras que las del sistema no; y además, en las tareas de sistema aparece la imagen de un óvalo inscrito en un rectángulo.

Ilustración 46: Tarea del sistema

Las variaciones de la navegación (**Modelo del Comportamiento**), se expresan con la etiqueta "Ir a…" asociada a una tarea de sistema o del usuario.

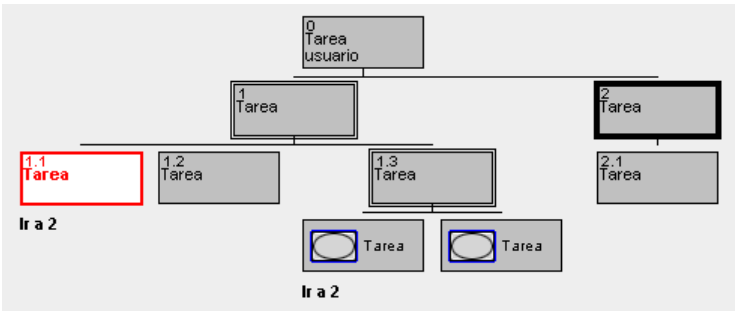


Ilustración 47: Ejemplo tarea de usuario y tarea del sistema con destino ("Ir a...")

En la figura 56 se puede observar que en dos casos, uno correspondiente a una tarea de usuario y otro correspondiente a una tarea de sistema, se obliga al usuario a seguir con la Tarea 2.

En el modelo del diálogo es posible también añadir características asociadas con la apariencia de la interfaz de usuario. En WebDiagram podemos caracterizar dos modos diferentes en la visualización del grupo de tareas en el prototipo:

- En una nueva ventana
- Tarea en una nueva sección de la misma ventana

Para indicar estas características del modelo de representación se utilizan los siguientes símbolos:

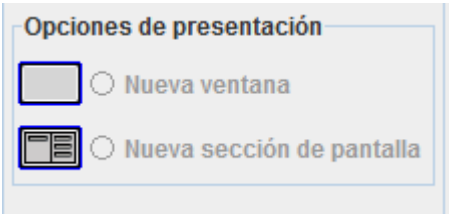


Ilustración 48: Opciones de presentación

Una vez realizado el modelo de diálogo, se puede generar automáticamente un prototipo para su evaluación. En éste se podrán observar los aspectos de interacción hombre-máquina, la navegación habitual y los caminos erróneos y aspectos de presentación en la interfaz. En WebDiagram 3.0, se han establecido unas **consideraciones genéricas (comunes a todos los objetivos desarrollados) en cuanto al prototipo,** agrupadas en aspectos de Navegación, Presentación y Comportamiento.:

- Navegación:
  - ✓ Normalmente existen varias secuencias correctas posibles
  - ✓ La navegación se considerará correcta si y solo si se efectúa una de las secuencias correctas consideradas en la especificación.
- Presentación:
  - ✓ El prototipo se lanzará en una ventana que podrá ajustarse en tamaño.
  - ✓ Únicamente habrá una ventana activa en el prototipo, salvo cuando haya tareas concurrentes.
  - ✓ El foco debe actualizarse en cada paso, y la activación de tareas debe señalar únicamente las posibles en cada momento.
  - ✓ No se van a limitar el número de secciones y desplegables posibles, aunque sí se lanzará aviso al llegar a tres acumuladas. Cuantas más secciones y desplegables se acumulen la visión será peor.
  - ✓ El diagrama irá señalando en rojo las tareas que se han ejecutado en el prototipo.
- Comportamiento:
  - ✓ En caso de simulación correcta (en presentación, navegación y comportamiento), el programa de simulación lanzará un aviso de finalización correcta. En caso contrario, el programa lanzará un aviso indicando que se ha cancelado el prototipo.
  - ✓ Si es posible no realizar más tareas en la ventana activa, ésta podrá cerrarse.

4.1.1.2 Características básicas y restricciones en los modelos de WebDiagram

A continuación se definen las características básicas de WebDiagram haciendo una descripción detallada de cada una de ellas, incluyendo las restricciones a tener en cuenta a la hora de realizar los diseños, de forma que no quede duda de cómo debe funcionar la herramienta a la hora de realizar los modelos y prototipos.

Las ***tareas de usuario*** se describen con dos características generales que se definen a continuación:

1.- **Orden**: Indica cuál es la secuencia de ejecución de las tareas.

2.- **Tipo**: Indica el número de veces que se puede realizar una tarea.

En función del ***ORDEN***, las tareas se clasifican en:

1.1.- **Secuencial**:

a) Se ejecutan todas las tareas.

b) Se ejecutan en orden numérico creciente.

1.2.- **Elección**:

a) Se ejecuta sólo una de las tareas elección hermanas.

b) No puede ser una tarea de tipo opcional.

1.3.- **Indiferente**:

a) Se ejecutan todas las tareas.

b) Se ejecutan en cualquier orden.

En función del TIPO, las tareas se clasifican en:

2.1.- **Unitaria**:

a) Se ejecuta una vez.

2.2.- **Opcional**:

a) Se ejecuta 0 o 1 veces.

2.3.- **Repetitiva**:

a) Se puede ejecutar 0, 1 o varias veces en función de los límites establecidos.

b) Debe tener al menos una hija, es decir, debe ser una tarea con hijas (Tarea objetivo).

c) El límite inferior debe ser menor o igual al límite superior.

d) Si el límite inferior es 0: se puede continuar sin ejecutar la tarea o ejecutarla tantas veces como el usuario quiera hasta un máximo de veces indicado por el límite superior.

e) Si el límite inferior es 1 o más: se ejecuta como mínimo tantas veces como indique el límite inferior y como máximo el número de veces indicado por el límite superior.

f) Si el límite inferior es igual que el límite superior: se ejecuta exactamente el número de veces que indique el límite.

En cuanto del ***Modelo de Comportamiento***, las tareas de usuario dependen de la etiqueta “Ir a…”:

a) La tarea seleccionada debe ser una tarea hoja.

b) La terea destino de “Ir a…” debe ser una tarea objetivo, es decir, una tarea con hijas.

c) Tras ejecutar una tarea con un “Ir A” aparecerá el mensaje “Efectuando la tarea *destino*”.

Tareas de sistema

Las tareas de sistema implican variaciones de la navegación y corresponden al modelo de comportamiento.

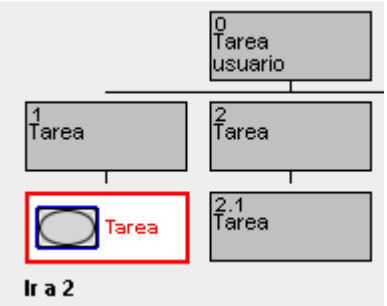


Ilustración 49: Tarea de sistema con variación de navegación (Ir a...)

Las variaciones se expresan con la etiqueta “Ir a…” asociada a la tarea de sistema.

Ilustración 50: Etiqueta "Ir a..."

a) Las tareas de sistema deben ser siempre tareas en nodos hojas de la jerarquía

b) Cuando hacemos variaciones en la navegación mediante la etiqueta “Ir a…”, la tarea destino debe ser una tarea objetivo; es decir, debe tener hijos.

c) A las tareas de sistema no se les puede asociar ninguna de las características de las tareas de usuario (orden y tipo).

d) No tienen características de presentación (nueva ventana, desplegar, nueva sección) porque no tienen tareas hijas.

e) Si a la tarea de sistema se le asigna un destino con la etiqueta “Ir a…”, los mensajes que escribimos en el cuadro de texto de mensajes (ver Ilustración 49) se mostrarán en una ventana cuando ejecutemos dicha tarea de sistema en el prototipo de diálogo. De esta forma, se informa al usuario sobre un comportamiento erróneo o desviado.

Características de presentación

Para establecer algunos aspectos sobre la apariencia de la interfaz de usuario, WebDiagram permite varias opciones de presentación.

Las diferentes formas de presentación se reflejarán en los prototipos del modelo de diálogo, de forma que veremos cómo la navegación por el prototipo es diferente en función de la característica de presentación asociada a cada tarea.

a) La tarea marcada con una opción de presentación marcará la forma de visualización de sus descendientes en la interfaz.

b) Para dar a una tarea una característica de presentación debe ser una tarea objetivo, es decir, debe tener hijas.

c) Tras la ejecución de una tarea en nueva sección, esta nueva sección desaparece.

d) El número de secciones está limitado a la resolución del a pantalla.

En WebDiagram 3.0 hay dos características diferentes sobre la visualización del grupo de tareas. A continuación se describen las dos opciones posibles:

- **Nueva ventana**: Es la opción por defecto y muestra las tareas hijas en una nueva ventana. Se representa mediante un trazo sencillo.

- **Tarea en una nueva sección de la misma ventana**: Muestra a las hijas a la derecha de la tarea padre, desplegadas en una nueva sección horizontalmente. Se representa mediante un trazo doble.

A continuación vemos un ejemplo que contiene todas las opciones de presentación:

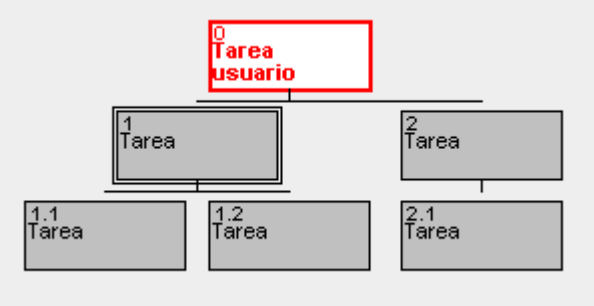


Ilustración 51: Ejemplo: Diferentes opciones de visualización

En la Ilustración 61 podemos seguir la ejecución del prototipo de diálogo y cómo se muestran las diferentes opciones de presentación del ejemplo presentado en la Ilustración 60.

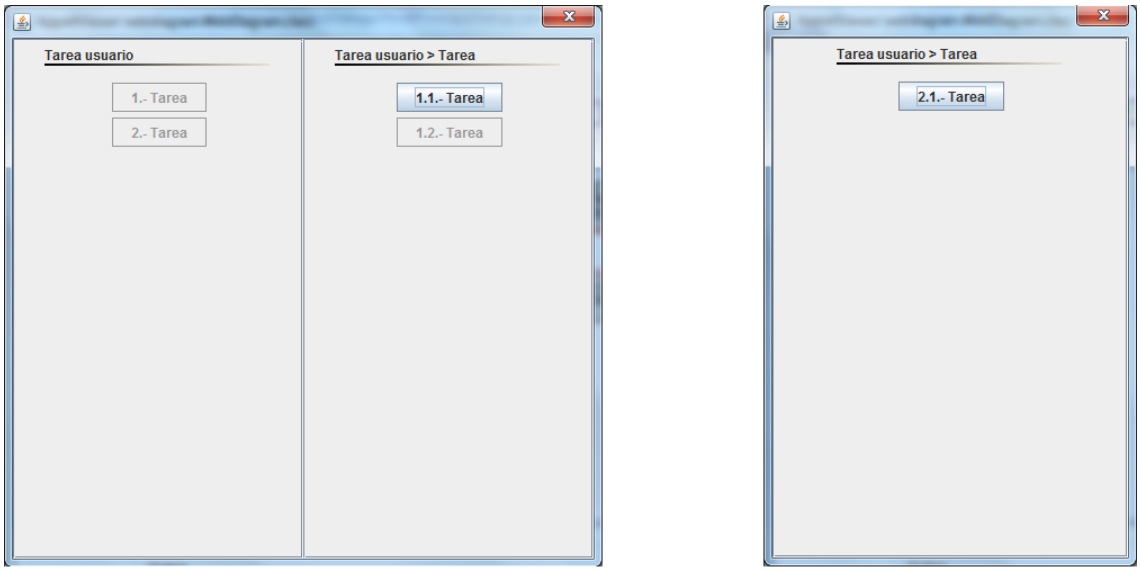


Ilustración 52: Ejemplo: Prototipo de diálogo con las diferentes opciones de visualización

4.1.1.3 Otras características y restricciones en WebDiagram

A continuación se tratan los casos especiales que no quedan recogidos en los apartados anteriores. Además se analizan las características complejas de WebDiagram y sus restricciones.

Tarea raíz

Este caso se da al comenzar un nuevo proyecto y no se podrán realizar las siguientes acciones:

a) Asignar ninguna propiedad a la tarea: orden, tipo, representación, etc.

b) Crear tareas hermanas.

c) Eliminar o mover la tarea.

d) Convertir la tarea en tarea de sistema.

e) Generar el prototipo de tareas ni el prototipo de usuario ya que no tiene subtareas.

f) Desglosar la tarea

El usuario únicamente podrá:

a) Cambiar el nombre de la tarea.

b) Añadir comentarios a la tarea.

c) Crear tareas hijas.

Tareas desglosadas:

Es posible particionar un diagrama o modelo en submodelos, para ello una tarea se caracteriza como tarea desglosada y aparecerá en una nueva pestaña como raíz de un subárbol del árbol principal.

a) No hay límite para el número de tareas que se pueden desglosar en otra pestaña del mismo nivel; es decir, la anchura no tiene límite, mientras que en profundidad, se pueden abrir como máximo tres tareas desglosadas una dentro de otra.

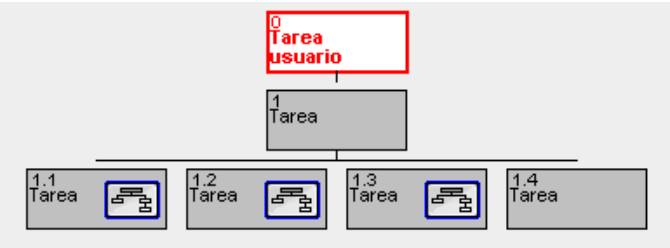


Ilustración 53: Tareas desglosadas

Ilustración 54: Tarea 1.3 desglosada

b) No se puede acceder mediante la etiqueta “Ir a…” a tareas hijas de una tarea desglosa.

c) Una tarea desglosada no podrá acceder mediante la etiqueta “Ir a…” a ninguna tarea objetivo, ya que en el momento que se marca como tarea desglosada, deja de considerarse una tarea hoja y pasa a ser una tarea padre.

d) En el diagrama de tareas en la que se encuentra la tarea desglosada, aparecerá un icono en dicha tarea indicando el desglose y no se podrán crear hijos de esa tarea en esa pestaña. Para crear hijos habrá que ir a la nueva pestaña creada con el desglose y crearlos en ella.

e) Una tarea desglosada puede volver a su situación original (sin desglose) seleccionando la tarea y eligiendo la opción de desglosar.

f) A partir de la versión WebDiagram 3.0, cada tarea desglosada o subárbol tendrá su propio subprototipo.

Tareas de orden concurrente:

Este es un Objetivo de Usuario nuevo en WebDiagram 3.0. Su funcionalidad, presentación y diseño de la navegación han sido definidos desde este punto.

Una tarea concurrente implica su realización en paralelo con sus tareas hermanas. Se muestran aquí sus restricciones a nivel de Diagrama y de Prototipo:

Diagrama:

a) En el diagrama o modelo aparece caracterizada con el símbolo: |||.

b) Una tarea concurrente no es de ningún orden ni tipo. Estas opciones aparecerán desactivas.

c) Una tarea concurrente no puede ser del Sistema. Es siempre tarea de Usuario.

d) Es posible desglosar una tarea concurrente

Prototipo:

a) En el prototipo, una tarea concurrente aparece como un botón verde que cuando es pulsado genera una ventana separada de color verde.

b) Los descendientes de la tarea concurrente son también concurrentes y aparecerán en la pantalla verde inicial. El color verde ira atenuándose a medida que la rama desciende.

c) Una tarea concurrente podrá ser pulsada en cualquier momento (no tiene orden) y el número de veces que se desee (no tiene tipo).

d) Una vez ejecutada una tarea concurrente, las subtareas correspondientes se podrán ejecutar de forma concurrente con el hilo principal u otras concurrentes.

e) Cuando hay varias concurrentes anidadas, deben cerrarse en orden inverso.

4.1.1.4 Otros requisitos provocados por las integraciones:

Requisitos del IR A (modelo de comportamiento con las tareas concurrentes):

a) La línea concurrente desde la que se realiza un “Ir A” (tenga hijas o no) se denomina como **origen** del salto y se tendrá que cerrar siempre.

b) Si el salto “Ir A” es hacia una concurrente, se ejecutará una nueva línea concurrente (nueva ventana verde) por la que se desviará el flujo de la ejecución. Las demás concurrentes abiertas se mantendrán intactas. El padre de la concurrente también estará visible, ya que se pueden ejecutar en paralelo.

Requisitos del IR A con las tareas no concurrentes:

a) Al hacer un "Ir A" a una indiferente, las demás mantendrán su estado (ejecutado o no ejecutado) salvo la origen que será cerrada. En caso de ser secuenciales, se marcarán como ejecutadas las anteriores al destino, las demás estarán por hacer y se harán en orden.

b) Cuando se hace un "Ir A" a una repetitiva, el número de ejecuciones de esa tarea se reiniciará a 0 automáticamente en lugar de incrementarlo.

Requisitos de concurrentes con modos de presentación

a) Las secciones que tienen tareas concurrentes pendientes (no concurrentes), únicamente desaparecen al pasar a otra activación de la sección anterior.

4.1.2 Requisitos no-funcionales

Esta aplicación no necesita una capacidad de memoria local donde deba ser alojada. Sin embargo, el hecho de ser una aplicación web que se ejecuta mediante navegadores como *Internet Explorer o Mozilla Firefox*, requiere la instalación de la maquina virtual de *Java* (*Java Runtime Environment*) en su versión 1.5 o superior en la máquina local.

En cuanto al servidor que alojará la herramienta, necesitará alrededor de 1MB de espacio libre en el disco. Al ser una aplicación que va a ser ejecutada a través de internet, no hay que olvidar el requisito de ser una herramienta ligera.

4.2 Ejemplo de un modelo de diagrama y su prototipo

WebDiagram es una aplicación web basada en *Diagram*, cuya función es facilitar la ejecución de algunas fases de la metodología *InterMod*. Para ello permite definir requerimientos, describir la relación entre el usuario y el sistema, y finalmente evaluar prototipos.

En **WebDiagram 3.0** se han arreglado numerosos errores heredados de la versión anterior, pero el objetivo principal ha sido lograr la ejecución del prototipo de diálogo y asegurar su funcionamiento para el diseño realizado por el usuario. A continuación se muestra un diagrama de ejemplo con el funcionamiento de su prototipo de diálogo.

4.2.1 Diagrama

En este ejemplo se muestra una aplicación que podrí servir para gestionar los libros de una biblioteca. En la pestaña principal se muestra el diagrama completo.

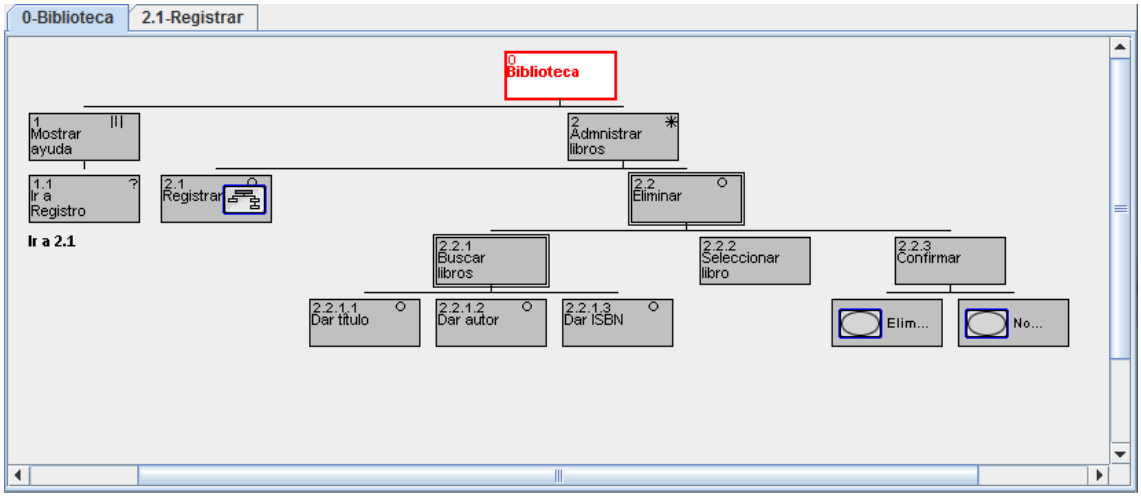


Ilustración 55: Diagrama de ejemplo, Biblioteca

En esta imagen se pueden apreciar los distintos elementos usados para lograr la aplicación deseada. Podemos encontrar tareas de diferentes tipos y orden, tareas del sistema, distintos modos de presentación e incluso una tarea desglosada.

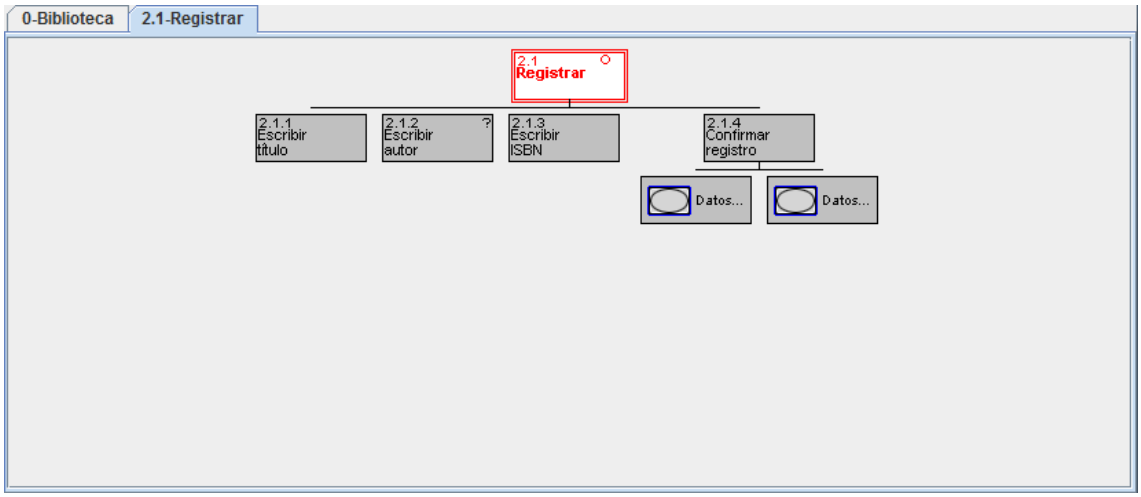


Ilustración 56: Tarea "Registrar" desglosada

4.2.2 Prototipo de diálogo

Para lanzar el prototipo de diálogo hay que acceder a la barra de menú y seleccionar la opción correspondiente. De todos modos también se puede usar un atajo con las teclas *Ctrl+D*. El prototipo se iniciará sobre la pestaña seleccionada, esto da la posibilidad de iniciar prototipos independientes en tareas desglosadas.

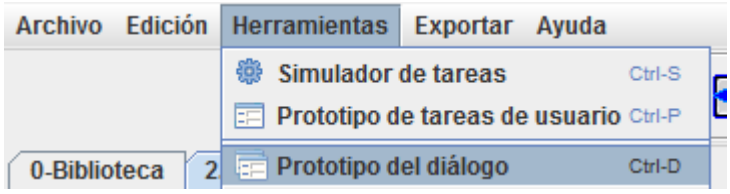


Ilustración 57: Menú de acceso al prototipo de diálogo

Una vez lanzado este es el aspecto de la ventana principal que se obtiene. Al tener en ella una tarea concurrente que se abre en una nueva ventana, podemos ejecutarla siempre que queramos.

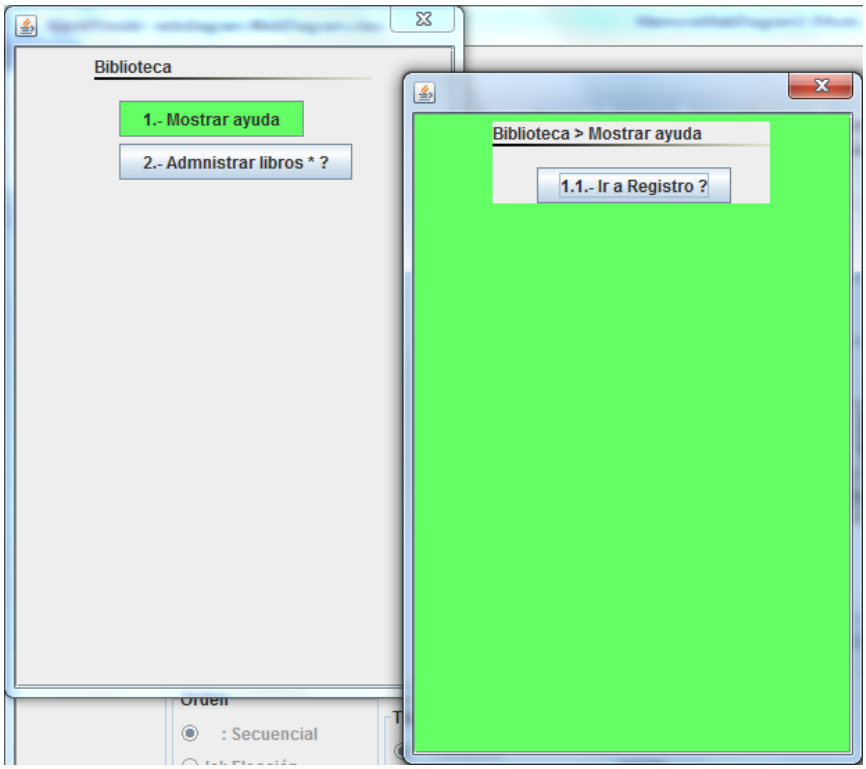


Ilustración 58: Prototipo de ejemplo I

Continuamos y accedemos a realizar la tarea “Administrar libros” que nos llevará a una nueva ventana que nos muestra dos opciones. La primera de ellas nos guiará por las tareas necesarias para registrar un nuevo libro, la otra tarea nos ayudará a eliminar un libro. Las dos opciones se despliegan en una nueva sección para tener constancia de donde venimos.

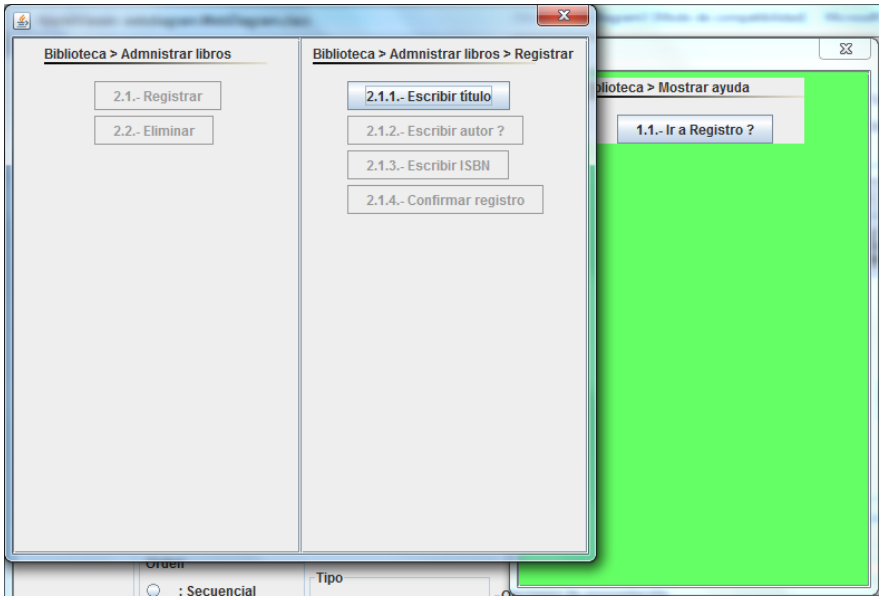


Ilustración 59: Prototipo de ejemplo II

En esta situación se puede ver que nos obliga a pasar por una serie de tareas para poder registrar el libro. La tarea "Escribir autor" es opcional porque puede haber libros sin autor, por lo que es posible no hacerla y pasar a la siguiente tarea.

Cuando llegamos a la tarea "Confirmar registro", se puede ver que nos lleva a una nueva ventana con dos tareas del sistema.

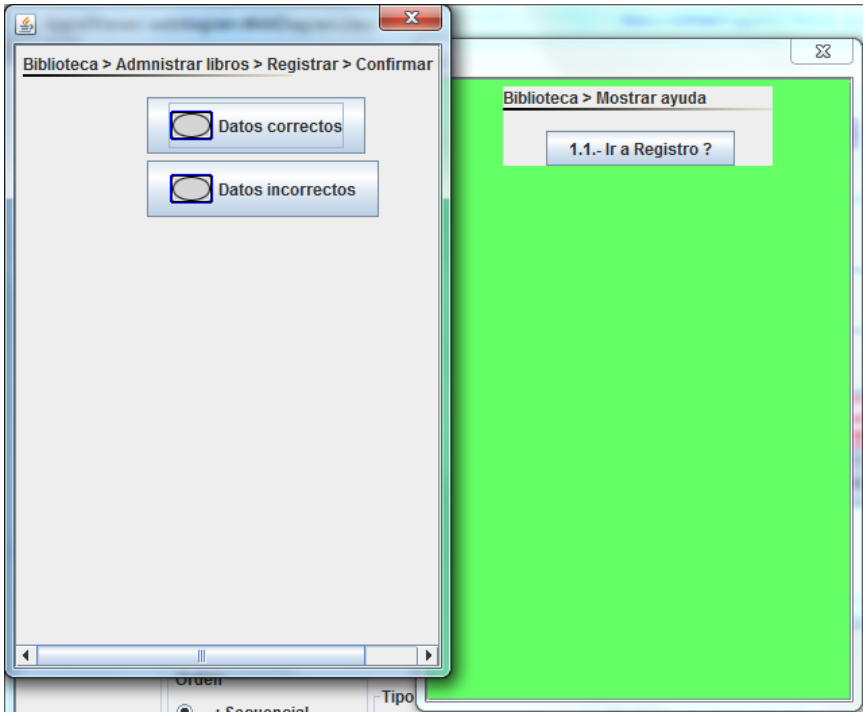


Ilustración 60: Prototipo de ejemplo III

Aquí se puede simular la interacción con la maquina y podemos decidir el resultado. En estos dos casos se nos mostrará un mensaje de información, al pulsar sobre la tarea "Datos correctos" se obtiene un mensaje confirmando el registro.

Ilustración 61: Mensaje del sistema

Después de aceptar el mensaje, el prototipo nos llevará de vuelta a la situación reflejada en la primera imagen. Si se hubiera pulsado en la tarea “Datos incorrectos” el mensaje hubiese sido distinto, pero llegaríamos a la misma situación, la ventana principal.

Durante toda la ejecución del prototipo se puede apreciar que tenemos visible la ventana concurrente. Esta ventana contiene una tarea opcional que en caso de ejecutarla se realiza un salto (Ir A) a la tarea “Registrar libro”, no importa donde esté la ejecución del prototipo.

Así que al pulsar la tarea “Ir a Registro” se cerrarán todas tareas abiertas y nos ejecutará automáticamente la tarea “Registrar” abriendo las ventanas y secciones necesarias.

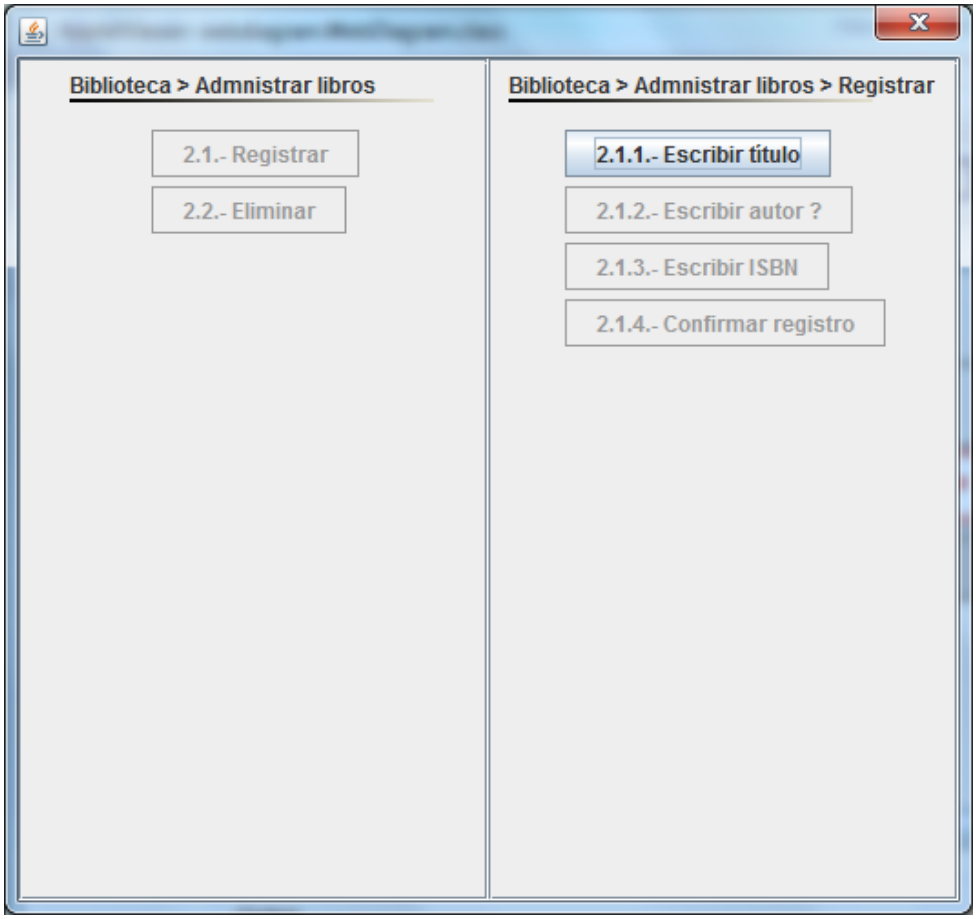


Ilustración 62: Prototipo de ejemplo IV

Una vez se termina de realizar esta tarea y al volver a la ventana principal, se puede cerrar el prototipo ya que las dos tareas no son obligatorias. Una de ellas es concurrente y la otra se ejecuta en un rango de 0 a todas la veces que se quiera.

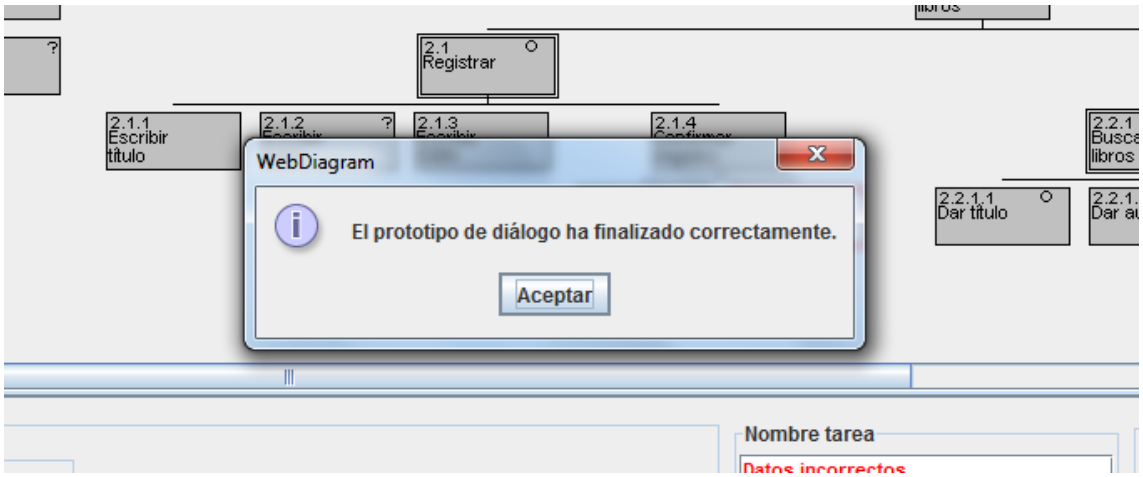


Ilustración 63: Finalización del prototipo de ejemplo

4.3 Diagrama de clases

Para poder entender como funciona desde el punto de vista del desarrollador se va mostrar el diagrama de clases correspondiente a esta versión. Dada la complejidad que podría llegar a tener, solamente se mostrarán las clases que han sido modificadas o usadas y se prescindirá de todas aquellas que no han tenido relación con el objetivo del proyecto.

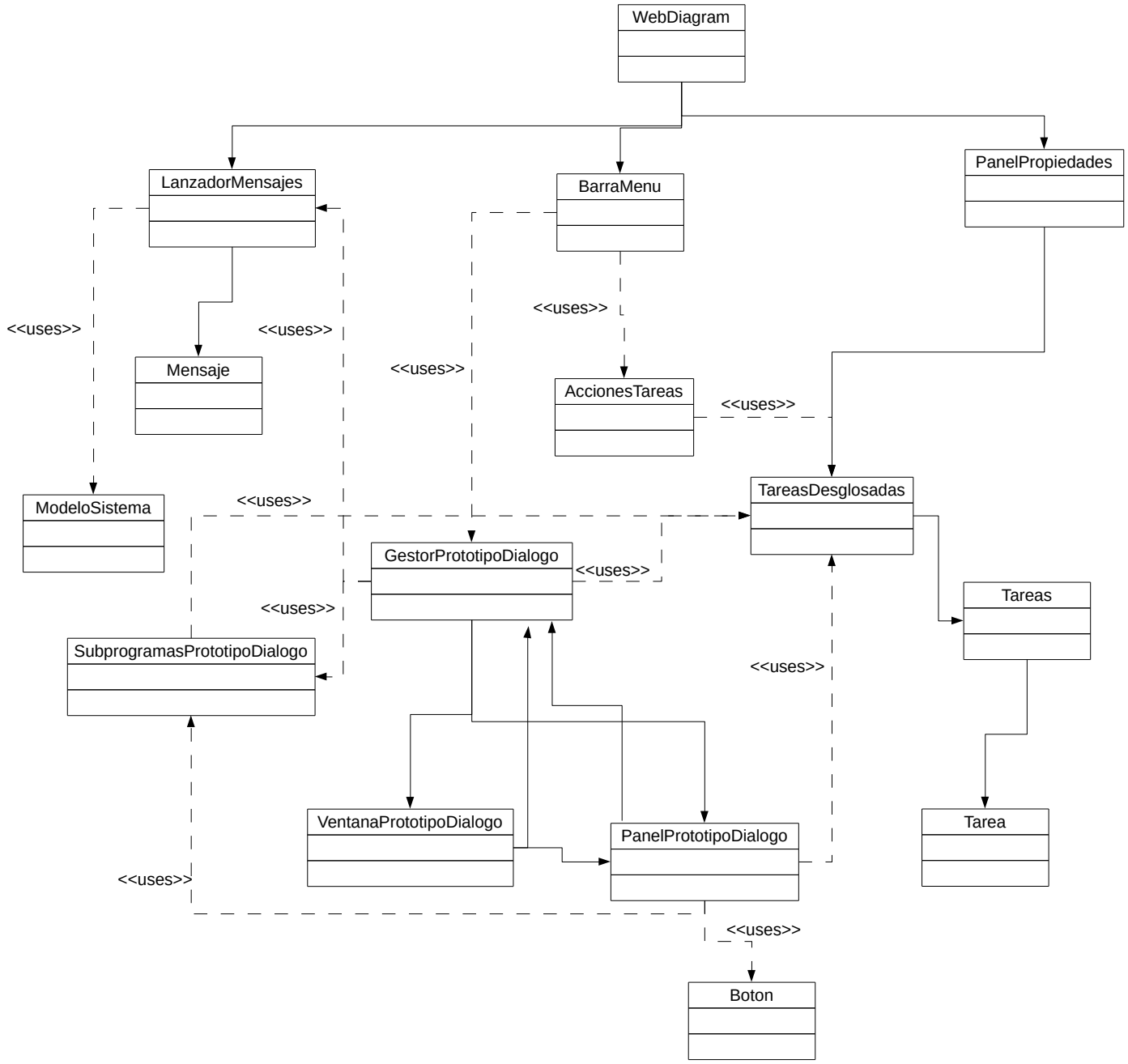


Ilustración 64: Diagrama de clases

4.4 Cambios en el código de WebDiagram 2.0

El cambio más destacado ha sido la eliminación de todas las clases relacionadas con el prototipo de diálogo anterior, ya que el desarrollo del mismo se ha realizado partiendo desde cero. Pero además de lograr un nuevo prototipo de diálogo funcional, se han ido corrigiendo numerosos errores en la lógica de negocio de WebDiagram ajenos al prototipo de dialogo:

- El programa no permitía tener la casilla “Ir A” activa a la tarea 1. Para arreglar esto se tuvo que eliminar una línea de código en la clase *PanelPropiedades*.

- Se decidió deshabilitar la opción de repetitiva para las tareas sin hijos, ya que la ejecución de las mismas una y otra vez no aparentaba cambio alguno.

- Con una única tarea en el diagrama, al lanzar el prototipo de diálogo no saltaba el aviso que nos informaba sobre el cambio al modelo de diálogo. Esta comprobación se realiza en la clase *GestorPrototipoDialogo* y al haber partido desde cero su desarrollo no hubo problemas para arreglarlo.

- Se vio que había muchos fallos en el tratamiento del mínimo y máximo de ejecuciones en el panel de propiedades. Los valores máximos podrían llegar a tener un valor más pequeño que el mínimo. Además se decidió que las tareas opcionales no podían tener un mínimo de 0. Se tuvieron que realizar todos los cambios necesarios en la clase *PanelPropiedades* en los “listeners“ de los “spinners”.

- A la hora de implementar la concurrencia se decidió no clasificarlo como un orden, así que se tuvo que cambiar la interfaz del panel de propiedades para habilitar un nuevo *JCheckBox* que permite habilitar o deshabilitar la concurrencia.

- Fue un cambio importante la inclusión de las nuevas clases LanzadorMensajes y Mensaje, los cuales gestionaban los mensajes lanzados en todo el programa y así poder traducirlos fácilmente en un futuro.

- Al eliminar hijas de una tarea, se decidió dejar esta tarea padre en unitaria. De lo contrario podría acabar siendo repetitiva y sin hijos, una situación no viable teniendo en cuenta decisiones anteriores.

- Se observó que faltaban ciertos métodos interesantes en la clase *TareasDesglosadas*, que finalmente fueron usados en el prototipo de diálogo. Se añadió un método llamado *getPestañaDandoIdTarea* que dando el identificador de cualquier tarea, nos devuelve la pestaña en la que se encuentra.

- Cuando se creaba una hermana a una tarea concurrente, esta nueva tarea cogía las propiedades de la hermana y se creaba como una concurrente. Se hicieron cambios en la clase *AccionesTareas* para cambiar el orden de esa nueva tarea para que fuera del orden de cualquier hermana no concurrente.

- Igual que pasaba con las tareas repetitivas, al eliminar las hijas de una tarea con un modo de presentación, ésta mantenía el modo de presentación. Una tarea sin hijas no puede tener modo de presentación, por lo que se tuvo que deshabilitar esta opción en la clase *PanelPropiedades*.

- Se cambió la clase *ModeloSistema* para que albergara el valor del límite máximo de secciones, pero esta vez calculado de forma dinámica dividiendo la resolución de pantalla entre el ancho de ventana.

- Otro método añadido en la clase *TareasDesglosadas* fue el usado para obtener una tarea dando su código en lugar del identificador. Es el método *getTareaDandoCodigo* que fue muy usado durante el desarrollo del prototipo.

- Un último cambio fuerte se hizo en la casilla “Ir A” del panel de propiedades. Los valores no se actualizaban correctamente cuando se eliminaban o se movían tareas. Por eso se hicieron cambios en la clase *PanelPropiedades* para guardar el estado de las tareas “Ir A” antes de realizar cada acción (nueva tarea, borrar tarea, mover tarea) y poder actualizar los valores posteriormente.

5. Conclusiones y líneas futuras

En este capítulo se analizan todas las tareas que han sido necesarias para completar los objetivos presentados en el capítulo 2. Se muestra una comparativa entre las horas estimadas y las horas reales invertidas que terminan con conclusiones generales y personales acerca de este proyecto. Finalmente, se sugieren posibles mejoras y arreglos que se podrían hacer a WebDiagram en futuros proyectos.

5.1 Objetivos alcanzados

A lo largo de la memoria se han ido mostrando todas las pruebas y todas las decisiones tomadas en las iteraciones. En cada una de ellas se han hecho cambios importantes con el fin de alcanzar los objetivos especificados en el capítulo 2. Gracias a todos estos cambios se ha conseguido un prototipo de diálogo funcional y una nueva versión de *WebDiagram* con numerosos errores corregidos.

Por si esto fuera poco, la creación del prototipo de diálogo ha requerido replantear y especificar el comportamiento de algunas funcionalidades que hasta ahora habían dado problemas. Entre ellas la funcionalidad "Ir A", los modos de presentación y las tareas concurrentes.

- **Objetivo 1: Lograr el correcto funcionamiento de un prototipo de diálogo siguiendo un desarrollo guiado por pruebas.**
  Se ha conseguido crear un prototipo de diálogo funcional. Para lograrlo, en cada iteración se ha tenido que analizar los resultados de las pruebas y se han tenido que tomar decisiones clave tanto de diseño como de implementación.
- **Objetivo 2: Gestionar el proyecto mediante el uso de una metodología ágil.**
  La gestión y planificación del proyecto se ha realizado de forma satisfactoria siguiendo las reglas de una variación de la metodología *Scrum*. También se ha seguido un desarrollo guiado por pruebas mediante la metodología *Test-driven Development*. Gracias a él, se han podido detectar fallos en cada iteración y encaminar el desarrollo corrigiéndolos uno a uno.
- **Objetivo 3: Reparar y mejorar los aspectos de WebDiagram2.0.**
  Además de lograr un prototipo de diálogo, se han corregido con creces los errores encontrados en la anterior versión de *WebDiagram*. En ocasiones se han tenido que corregir ciertos fallos generados por causa de los continuos cambios en el diseño. Cambios totalmente permisibles gracias al uso de la metodología ágil.

5.2 Comparativas entre la estimación y horas invertidas

En esta sección se muestra una comparativa entre las horas totales estimadas en el *Sprint Backlog* inicial y las horas reales invertidas en el proyecto. En la primera tabla se puede apreciar exactamente la diferencia entre las horas estimadas y las horas invertidas. Cabe destacar que en algunas de las tareas se han invertido más horas de las estimadas, de hecho

en cada iteración se han ido reajustando las estimaciones para amoldarlas a las necesidades del proyecto.

Fases	Tareas			Horas totales estimadas	Horas invertidas
Gestión	Reuniones	Preparación del orden del día		1:00	1:35
		Realizar la reunión		15:00	37:00
		Recoger el acta		3:00	4:35
	Planificación inicial			5:00	4:20
	Realizar el Sprint Backlog			6:00	12:40
	Archivo	Mantenimiento		1:00	0:50
		Copias de seguridad		2:00	2:40
Formación	WebDiagram			4:00	5:10
	Librerías gráficas			6:00	3:15
	Patrones Java			2:00	0:50
	Metodologías ágiles			4:00	2:00
Instalación	Instalación de las herramientas necesarias			2:00	3:00
Captura de requisitos	Casos de uso			1:00	0:30
	Prototipo de la interfaz			2:00	2:15
Diseño	Diseño de la lógica de negocio			20:00	17:00
	Diseño del modelo de presentación			10:00	5:50
Implementación	Definir la arquitectura			1:00	0:45
	Implementación de la interfaz			3:00	4:00
	Implementación de la lógica de negocio		Tareas de orden secuencial	6:00	7:10
			Tareas de elección	7:00	8:50
			Tareas de orden indiferente	10:00	3:15
			Tareas de orden concurrente	7:00	12:00
			Presentación	9:00	12:25
			Tareas del sistema	2:00	0:30

Fases	Tareas		Horas totales estimadas	Horas invertidas
		Funcionalidad "Ir A"	6:00	14:40
Pruebas	Diseño de pruebas		10:00	0:00
	Ejecución de pruebas		15:00	10:15
Documentación	Desarrollo de la memoria		60:00	39:50
	Creación del manual de usuario		2:00	1:00
	Elaboración de la presentación		15:00	0:00
TOTAL			237:00	218:10

Tabla 26: Horas totales estimadas vs Horas invertidas

Se puede ver que la diferencia total entre las horas estimadas y las horas invertidas no es tan grande. Pero si se analiza cada tarea, la cosa cambia.

En un principio no se planteó hacer una **reunión** por semana, por lo que se estimaron 15 horas a las reuniones. Finalmente se ha ido aumentando la cantidad de horas cuando se veía que no eran suficientes.

Algo parecido ha ocurrido con la implementación de las **tareas concurrentes y la funcionalidad** "Ir A". Al principio se planteó implementar las bases sin pensar en los problemas que podían surgir. Más tarde se fueron tomando decisiones importantes en cada iteración acerca este tema y no hubo más remedio que aumentar las horas.

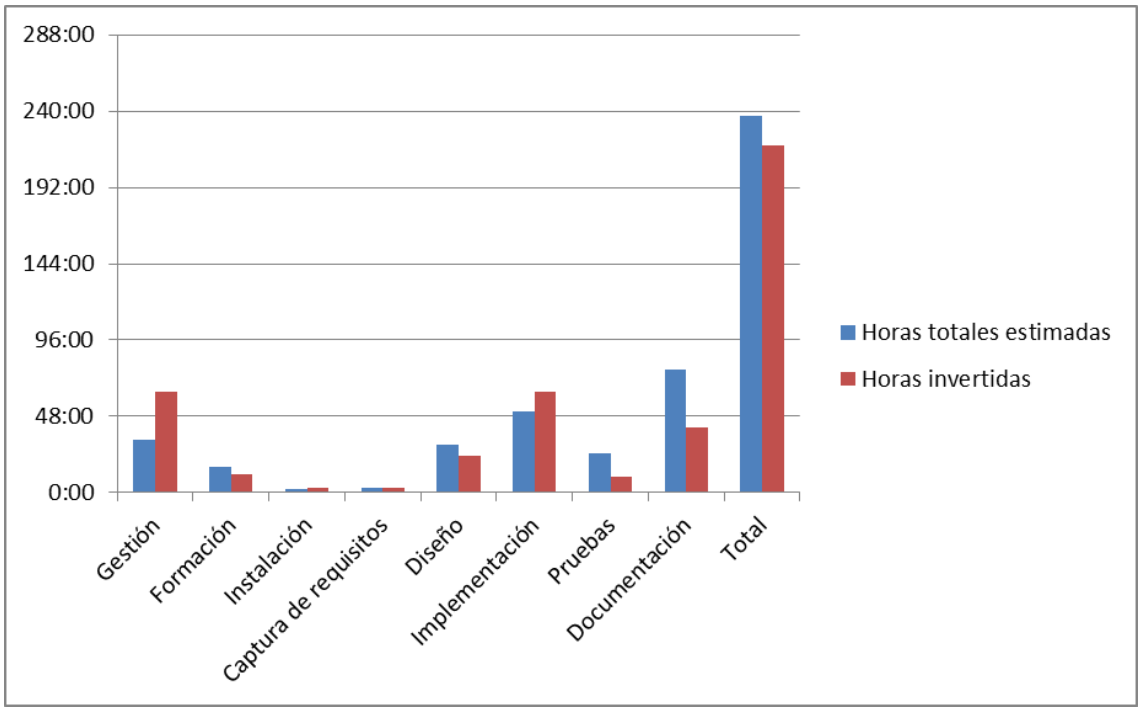


Ilustración 65: Gráfica de horas invertidas por fases

En el gráfico de arriba se puede ver la diferencia de horas clasificadas por fases. Se puede apreciar cómo en la fase de documentación hay una diferencia bastante notable entre las

horas estimadas e invertidas. Al haber seguido una metodología ágil para la gestión del proyecto, han aumentado las horas invertidas en la gestión y se han reducido las horas invertidas en la documentación. De hecho, la documentación en las metodologías ágiles no desempeñan un papel importante.

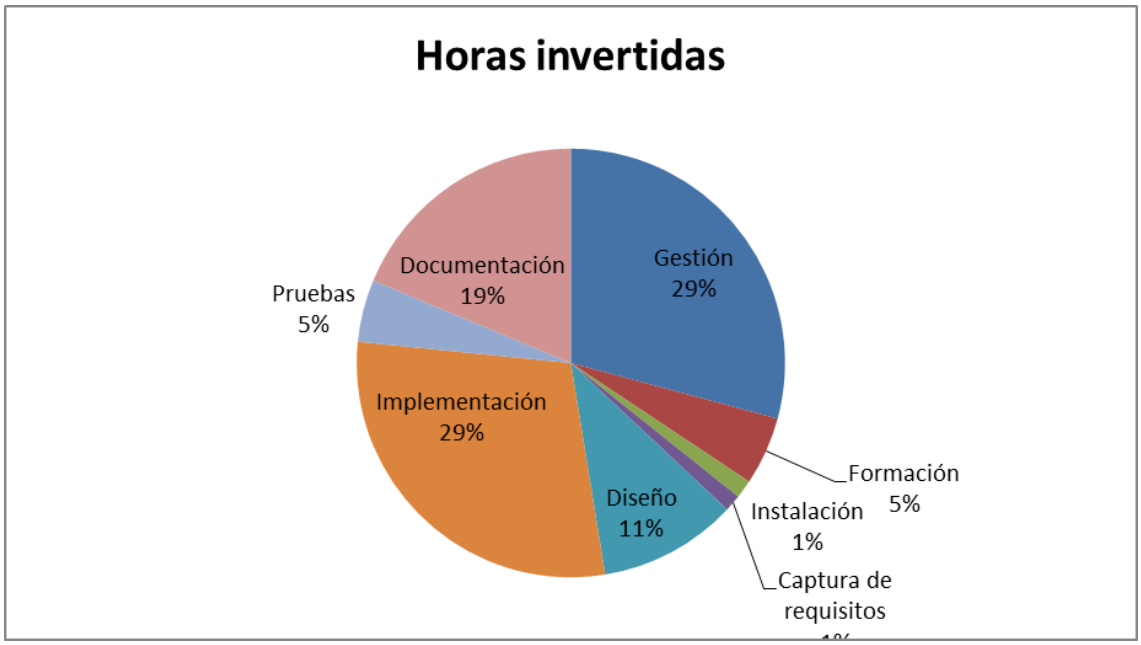


Ilustración 66: Relación de horas invertidas

Para finalizar, en la gráfica circular se puede ver que las fases de gestión e implementación son las que han predominado frente a las demás. Es un dato que mantiene toda la lógica, ya que los objetivos del proyecto especificados en el capítulo 2 se incluyen en esas fases.

5.3 Conclusiones

Se presentan las conclusiones generales y las personales, dejando constancia de lo que ha supuesto el desarrollo del proyecto.

5.3.1 Generales

Como ya se ha visto en e apartado anterior, tras la finalización del proyecto puede concluirse que se han cumplido todos los objetivos propuestos.

Se ha conseguido una nueva versión de *WebDiagram* más funcional con un prototipo de diálogo que mantiene la apariencia pero aumenta sus capacidades. Durante la implementación del prototipo de diálogo se han creado pocas clases y se ha programado de manera ordenada para que sea mucho más fácil comprender su funcionamiento.

Las nuevas especificaciones en el diseño de las tareas concurrentes y la funcionalidad “Ir A” han dado como fruto un programa más completo. Se han mantenido dos modos de presentación y se ha aparcado el desarrollo del modo de presentación en desplegables. Esta podría ser una posible mejora mirando hacia el futuro de *WebDiagram*.

5.3.2 Personales

Este proyecto ha sido una gran experiencia a nivel personal. El interés sobre las metodologías ágiles y la escasa experiencia sobre el trabajo en equipo han sido un buen punto de partida para llevarlo a cabo.

Gracias a la experiencia que he obtenido estos últimos años con la programación en C++, Java y la programación orientada a objetos en general, no he tenido problemas para entender el código de la versión anterior con rapidez.

Por otra parte, tengo que decir que el mantenimiento de toda la documentación del proyecto no ha sido fácil. Al no tener la costumbre de documentar los cambios programados ni tener experiencia a la hora de realizar estimaciones, ha sido una tarea pesada el tener que documentar extensamente cada *Sprint Backlog*.

De todos modos, el haber citado reuniones cada semana ha amenizado todo el proceso de gestión, documentación e implementación. Además, ha sido de gran ayuda haber contado en las reuniones con la experiencia de Begoña Losada y Juan Miguel López.

Por último, he podido comprobar la efectividad de la metodología *Test-driven Development* en un proyecto de tal envergadura con tantos cambios en cada iteración. Tengo que decir que el haber usado esta metodología para el desarrollo y una variación del *Scrum* para la gestión del proyecto, ha sido un gran acierto. Y teniendo en cuenta que este tipo de proyectos se desarrollan en equipo, puedo decir con toda seguridad que he adquirido una gran experiencia en esta área de cara al futuro.

5.4 Líneas futuras

En cuanto a las posibles líneas de actuación a partir de este proyecto de fin de carrera, se propone la siguiente:

5.4.1 Modo de presentación desplegable

Este nuevo modo de presentación ha quedado finalmente fuera de los objetivos de este proyecto. Durante la implementación del modo de presentación en secciones, se han programado todos los aspectos cuidadosamente teniendo constancia de que en un futuro pueda existir este nuevo modo de presentación.

Así, a la hora de implementar el modo de presentación desplegable no hará falta entrar en la implementación de la lógica de negocio, bastará con cambiar la forma de presentación de los paneles y los botones jugando con distintos Layouts de Java.

6. Referencias

[1] Losada, B., Urretavizcaya, M., Fernández-Castro, I.,. *Agile Development of Interactive Software by means of User Objectives.* 6th Int. Conf. on Software Engineering Advances, Barcelona. (2011).

[2] Losada, B., Urretavizcaya, M.,Fernández-Castro, I.: The InterMod Methodology: An Interface Engineering Process linked with Software Engineering Stages, In Macías,J.A., Granollers,T., Latorre,P.(eds). New Trends on Human-Computer Interaction: Reseach, Development, New Tools and Methods. Springer (2009).

[3] Beck, K., : Test-Driven Development, by example. Addison-Wesley, 2010

[4] Johansen, C : Test-Driven JavaScript Development,Addison-Wesley, Boston, 2010.

[5] Wikipedia. You ain't gonna need it . http://en.wikipedia.org/wiki/You_ain't_gonna_need_it. Último acceso julio 2012

Anexos

Anexo I
Manual para el desarrollador

Para cargar la aplicación en eclipse:

- Abrir Eclipse.
- Crear un nuevo proyecto de java (File>New>Java Project). Elegir un nombre para el proyecto y darle a Finish.
- NOTA: La aplicación se ha creado sobre el JDK 1.6, así que es recomendable utilizar dicha versión o una superior (puede funcionar con una inferior, pero no puede asegurarse). Se puede descargar dicha versión de la página oficial de Java.
- Seleccionar la opción "Import" en el menú File.
- En la pantalla que aparece, elegir "Archive File" (en la carpeta General) y pulsar Next.
- En la parte de "From archive file" pulsar "Browse".
- Seleccionar el archivo comprimido que contiene el código de la aplicación y pulsar "Abrir".
- Aparecerá (en uno de los recuadros que estaba en blanco) el sistema de ficheros que contiene el archivo comprimido. Ahí, seleccionar únicamente las carpetas "icons" y "webdiagram".
- En la parte de "Into Folder" pulsar "Browse", elegir la carpeta del nuevo proyecto que hemos creado y dentro de ella seleccionar la carpeta "src", pulsar OK.
- El proyecto se habrá cargado correctamente y ya se podrá trabajar sobre él o ejecutarlo.

Para ejecutar el proyecto desde eclipse:

- Hacer click derecho sobre la carpeta en la que hemos cargado el proyecto y elegir "Run As" y "Java Applet".

Para instalar la aplicación en un servidor:

- En Eclipse elegir la opción "Export" del menú "File".
- En la pantalla que aparece, elegir "JAR File" (en la carpeta Java). Pulsar Next.
- Seleccionar la carpeta que contenga el proyecto (asegurarse que la carpeta "src" está seleccionada").
- En la parte que indica "Select the export destination", elegir dónde se quiere guardar el archivo y asegurarse de nombrarlo "webDiagram.jar" *.
- Firmar el archivo que acabamos de exportar (ver Anexo II *"Firma de un*

- *Applet").*
- Colocar en el servidor los archivos webDiagram.jar (ya firmado), WD.html y webdiagram.jnlp (estos dos últimos están incluidos en el archivo junto al código de la aplicación).
- Acceder al archivo WD.html, WebDiagram debería desplegarse normalmente (primero pide permiso e informa de quién ha firmado el Applet).

* Si quiere cambiarse el nombre de webDiagram.jar, será necesario modificar el parámetro "name" del archivo webdiagram.jnlp (dicho archivo puede editarse con el bloc de notas) para que todo funcione correctamente.

Anexo II
Firma de un Applet

¿Por qué es necesario?

Como ya se ha mencionado en este documento, para que esta herramienta funcione correctamente en un explorador es necesario que el Applet esté firmado. La explicación de por qué es necesario este paso es simple: por seguridad. Un Applet por defecto no puede tener acceso a los recursos del ordenador que lo visualiza puesto que si fuera de otra manera podríamos encontrarnos ante Applets que, simplemente por ser visualizados, borran discos duros, acceden a ficheros personales, etc.

Pero es evidente que en casos como el que nos ocupa es necesario que el Applet tenga acceso a ciertos recursos del equipo, puesto que queremos que el usuario pueda, por ejemplo, guardar su trabajo en su propia máquina. Así pues existe un método para que el Applet pueda acceder a los recursos del equipo: firmar el Applet digitalmente. De esta forma, cuando se vaya a visualizar el Applet firmado, se preguntará al usuario si confía en la persona o entidad que ha firmado el mismo y, en caso positivo, se le dará los permisos necesarios para ejecutarse.

Proceso para realizar la firma

En primer lugar deberemos hacer los siguientes preparativos:

- Para firmar digitalmente un Applet es necesario tener instalado un kit de desarrollo de Java (Java Development Kip, JDK), no siendo suficiente disponer de un entorno de ejecución de Java (Java Runtime Environment, JRE). Podremos descargar el JDK de la página oficial de Java.
- Debemos conocer dónde se encuentra el JDK instalado. Por defecto suele encontrarse en "Archivos de Programa\Java". Anotaremos la ruta de dicha carpeta (por ejemplo, "C:\Program Files (x86)\Java\jdk1.6.0_18") a la que de ahora en adelante nos referiremos como RUTA_JAVA.
- Una vez se tiene el Applet en un archivo .jar, es necesario abrir la consola de comandos con permiso de administrador. Para ello, seleccionamos Inicio>Programas>Accesorios, hacemos click derecho sobre "Símbolo del sistema" y seleccionamos "Ejecutar como administrador".

En primer lugar deberemos crear la firma (este paso no es necesario repetirlo cada vez que queramos firmar un Applet):

- Escribimos en la consola el siguiente texto y pulsamos enter:

```
RUTA_JAVA\bin\keytool -genkey -alias nombreClave -validity 120 -v
```

donde "nombreClave" es un alias para esta clave y "120" será el número de días de validez que tendrá la clave.

- Se nos pedirán los siguientes datos personales:
  - ✓ Contraseña del almacén de claves (si no se ha creado ninguna firma anteriormente, habrá que elegir una contraseña que deberemos recordar).
  - ✓ Nombre y Apellido.
  - ✓ Nombre de la unidad de organización (departamento).
  - ✓ Nombre de la organización.
  - ✓ Nombre de la ciudad o localidad.
  - ✓ Nombre de la provincia.
  - ✓ Código del país de dos letras (*ES,* de España).
- Una vez rellenados los datos se nos mostrarán de nuevo en pantalla y se nos pedirá que confirmemos que son correctos.
- Por último se nos pedirá una contraseña para la clave. Una vez elegida, se creará la firma digital.

A continuación firmaremos el Applet deseado de la siguiente forma:

- En la consola de comandos escribimos el siguiente texto y pulsamos enter:

```
RUTA_JAVA\jarsigner.exe RUTA_APPLET\miApplet.jar nombreClave -verbose
```

donde:

- ✓ RUTA_JAVA: la ruta del JDK (como ya se ha mencionado antes)
- ✓ RUTA_APPLET: la ruta de la carpeta en la que se encuentra el Applet (cuidado, que no sea de sólo lectura).
- ✓ miApplet.jar: el nombre del Applet que queramos firmar.
- ✓ nombreClave: el alias de la firma que queramos usar.

- Se nos pedirá que introduzcamos la clave de la firma.
- Se muestra un aviso de que la validez de la firma será de X días (120 en el ejemplo).
- El Applet está firmado y listo para usarse.

Anexo III
Pruebas de evaluación

"TDD es un proceso de desarrollo iterativo en el que cada iteración comienza escribiendo un test que forma parte de la especificación que estamos implementando. Las iteraciones cortas permiten un feedback instantáneo en el código que estamos escribiendo, y las decisiones malas son fáciles de detectar".

Cada iteración en TDD consiste en los siguientes cuatro pasos:

- Escribir un test: tomar una característica a implementar, y escribir una unidad de test para él.
- Ejecutar los test; observar el fallo del nuevo test. Si no funciona, si se soluciona fácilmente, arreglarlo; si no, tomar nota. El fallo nos ayudará a confirmar nuestras teorías sobre el estado actual de nuestro código. Mientras se escribe el test, debe haber una expectación sobre cómo va a fallar el test.
- Hacer que pase el test. Utilizar la solución más simple que hace que se ejecute. Seguir el principio "you ain't gonna need it" o YAGNI, según el cual no se debe añadir funcionalidad hasta que no sea necesario.
- Refactorizar para eliminar duplicación. Revisar el trabajo y hacer ajustes para eliminar la duplicación y mejorar el diseño.

El objetivo es código limpio que funcione, en este orden:

1. Que funcione
2. Conseguir código limpio

En cada iteración el test es la especificación. Aunque no haya Big Design Up Front haciendo TDD, debemos invertir tiempo en algo de diseño antes de lanzar una sesión TDD: "Writing the test itself is an act of design" [4].

Las unidades de test en un desarrollo TDD son trozos de código que comprueban un resultado esperado en función de una entrada.

Las pruebas para WebDiagram 3.0 tienen entradas y salidas gráficas. A partir de una entrada gráfica (un diagrama de tareas) queremos comprobar varios comportamientos y presentaciones en un prototipo de tipo gráfico (ventanas, botones, secciones, colocación, número y orden de accesos, etc). Por lo tanto, el proceso tradicional de pruebas empleado en TDD con herramientas tipo jUnit no van a ser posibles.

La metodología de proceso que vamos a seguir es TDD, sin embargo las unidades de test no estarán programadas. Se realizará un proceso similar pero provocando la ejecución manualmente. Dada una figura, comprobaremos si los efectos coinciden con la especificación.

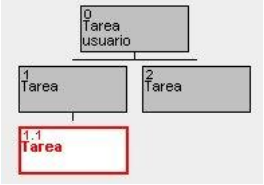
Consideraciones genéricas (comunes a todos los objetivos desarrollados):

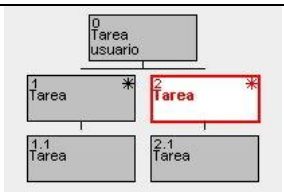
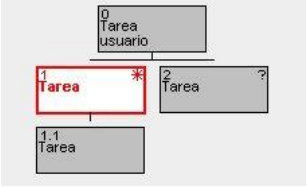
- **Presentación**:
  - El prototipo se lanzará en una ventana que podrá ajustarse en tamaño.
  - Únicamente habrá una ventana activa en el prototipo, salvo cuando haya tareas concurrentes.
  - El foco debe actualizarse en cada paso, y la activación de tareas debe señalar únicamente las posibles en cada momento.
  - No se van a limitar el número de secciones y desplegables posibles, aunque sí se lanzará aviso al llegar a tres acumuladas. Cuantas más secciones y desplegables se acumulen la visión será peor.
  - El diagrama irá señalando en rojo las tareas que se han ejecutado en el prototipo.
- **Navegación**:
  - Normalmente existen varias secuencias correctas posibles
  - La navegación se considerará correcta si y solo si se efectúa una de las secuencias correctas consideradas en la especificación.
- **Comportamiento**:
  - En caso de simulación correcta (en presentación, navegación y comportamiento), el programa de simulación lanzará un aviso de finalización correcta. En caso contrario, el programa lanzará un aviso indicando que se ha cancelado el prototipo.
  - Si es posible no realizar más tareas en la ventana activa, ésta podrá cerrarse.

Unidades de test y su efecto en el prototipo:

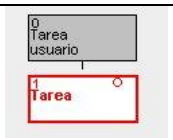
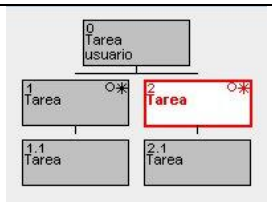
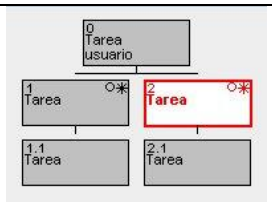
Unidades de Test Objetivo 1-tS: Tareas de orden secuencial

Nombre	Figura	Descripción	Efecto
TS_0		Una tarea principal	AVISO: No hay prototipo
TS_0		Una tarea principal con una hija unitaria	Una ventana con un botón. Se pulsa y finaliza correctamente Secuencia: 1
TS_1		Una tarea principal con una hija de tipo opcional	Una ventana con un botón. Se pulsa o se cierra la ventana y finaliza correctamente Secuencias: 1, x
TS_2	AVISO: No es posible porque la tarea repetitiva no tiene hijas.	Una tarea principal con una hija de tipo repetitivo	
TS_3		Varias hijas secuenciales, unitarias	Una ventana con tres botones. Hay que pulsarlos secuencialmente para finalizar correctamente. Secuencia: 1-2-3
TS_4		Varias hijas secuenciales; la primera, opcional	Una ventana con dos botones. Finalizará correctamente si se pulsan secuencialmente o si se pulsa únicamente el segundo. Secuencias: 1-2, 2

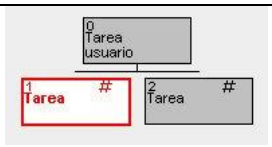
Nombre	Figura	Descripción	Efecto
TS_5		Varias hijas secuenciales, de tipo opcional	Una ventana con dos botones. Finalizará correctamente si se pulsan secuencialmente o si se pulsa uno de ellos o si se cierra la ventana. Secuencias: 1-2, 1, 2, x
TS_6		Varias hijas y nietas de orden secuencial, tipo unitario	Una ventana con dos botones a activar secuencialmente. El primero produce un botón que ha de activarse: 1-1.1-2
TS_7		Varias hijas y nietas de orden secuencial, una hija opcional	
TS_8		Varias hijas y nietas de orden secuencial, una hija repetitiva Min=Max=0	AVISO: Las repetitivas deben poder efectuarse al menos una vez. La deja en unitaria
TS_9		Varias hijas y nietas de orden secuencial, una hija repetitiva Min=Max=1	La convierte en unitaria Secuencia: 1-2
TS_10		Varias hijas y nietas de orden secuencial, una hija repetitiva Min=0, Max=999	Secuencias: (1-1.1)*- 2 2

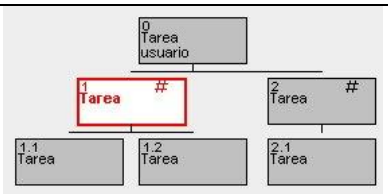
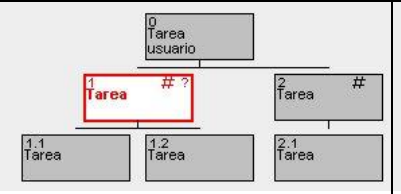
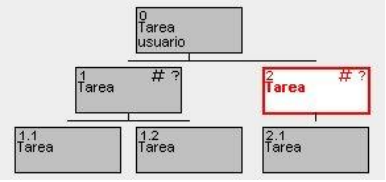
Nombre	Figura	Descripción	Efecto
TS_11		Todas las hijas repetitivas Min=0, Max=999	Secuencias correctas: (1-1.1)*- (2-2.1)* (1-1.1)* (2-2.1)* X
TS_12		Tareas de orden secuencial. Una hija repetitiva min=2, Max=4, la otra opcional	Secuencias correctas: (1-1.1)[2..4]- 2 (1-1.1)[2..4]
TS_13		Tareas de orden secuencial. Todas opcionales.	Secuencias correctas: 1-2-2.1-3 2-2.1-3 1-3 1 2 3 x
TS_14		Tarea secuencial, iterativa de 0 a 3 veces.	Secuencias correctas: (1-1.1)* máx 3 veces x

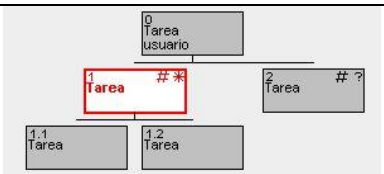
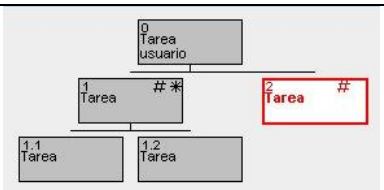
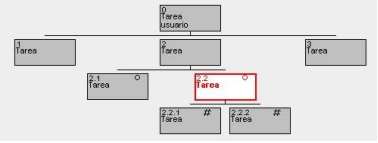
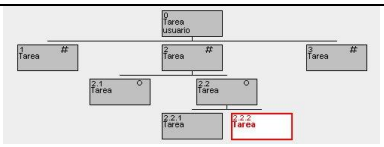
U.T. Objetivo 2-TSE: tS+Tareas de integración: orden elección + Secuenciales:

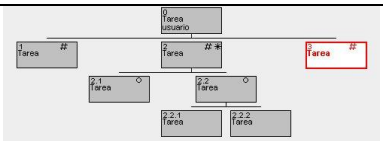
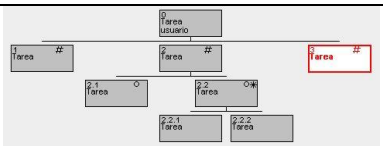
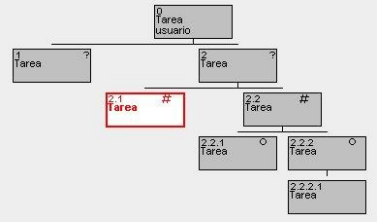
Nombre	Figura	Descripción	Efecto
TSE_1		Una tarea principal con una hija de orden elección y tipo unitaria	Una ventana con un botón. Se pulsa y finaliza correctamente. Secuencia: 1
TSE_2	AVISO: No es posible una tarea de orden elección y tipo opcional	Una tarea principal con una hija de orden elección y tipo opcional	
TSE_3		Varias hijas de orden elección, unitarias	Una ventana con dos botones. Finalizará correctamente si se pulsan únicamente uno. Secuencias: 1, 2
TSE_4		Varias hijas y nietas. Las hijas de orden elección y repetitivo, Min=0, Max=999	AVISO: Las tareas repetitivas de orden elección, su Min debe ser mayor que 0.
TSE_5		Varias hijas y nietas. Las hijas de orden elección y repetitivo, Min=1, Max=999	Secuencias correctas: (1-1.1)* (2-2.1)* X
TSE_6		Varias hijas y nietas de orden elección y unitarias	Secuencias correctas: 1-1.1-1.2 2-2.1
TSE_7		Varias hijas y nietas de orden elección y unitarias	Secuencias correctas: 1-2-2.1-3 1-2-2.2-3

U.T. Objetivo 3 tSEI: tSE+ Tareas de integración: orden Secuenciales + elección + Indiferentes:

Nombre	Figura	Descripción	Efecto
TSEI_1		Una tarea principal con una hija de orden indiferente y tipo unitaria	Una ventana con un botón. Se pulsa y finaliza correctamente. Secuencia: 1
TSEI_2		Una tarea principal con una hija de orden indiferente y tipo opcional	Una ventana con un botón. Se pulsa o se cierra la ventana y finaliza correctamente. Secuencias: 1, x
TSEI_3		Varias hijas de orden indiferente, unitarias	Una ventana con dos botones. Finalizará correctamente si se pulsan los dos botones (en cualquier orden). Secuencias: 1-2, 2-1
TSEI_4		Varias hijas de orden indiferente, una opcional	Una ventana con dos botones. Finalizará correctamente si se pulsan los dos en cualquier orden o si se pulsa únicamente el segundo. Secuencias: 1-2, 2-1, 2
TSEI_5		Varias hijas de orden indiferente, todas opcionales	Una ventana con dos botones. Finalizará correctamente si se pulsan en cualquier orden o si se pulsa uno de ellos o si se cierra la ventana. Secuencias: 1-2, 2-1, 1, 2, x

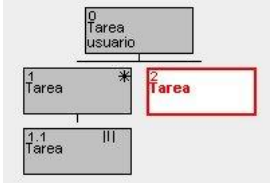
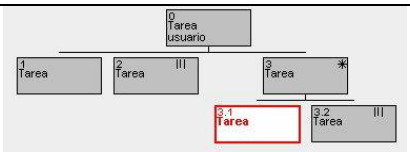
Nombre	Figura	Descripción	Efecto
TSEI_6		Varias hijas y nietas de orden indiferente y unitarias	Secuencias correctas: 1-1.1-1.2-2-2.1 2-2.1-1-1.1-1.2
TSEI_7		Varias hijas, y nietas, de orden indiferente y de tipo opcional y unitaria	Secuencias correctas: 1-1.1-1.2-2-2.1 2-2.1-1-1.1-1.2 2-2.1
TSEI_8		Varias hijas, y nietas, de orden indiferente y de tipo opcional	Secuencias correctas: 1-1.1-1.2-2-2.1 2-2.1-1-1.1-1.2 1-1.1-1.2 2-2.1 x
TSEI_9		Varias hijas, y nietas, de orden indiferente y de tipo repetitivo (Min=0, Max=999), y opcional	Secuencias correctas: (1-1.1-1.2)*-2-2.1 2-2.1-(1-1.1-1.2)* (1-1.1-1.2)* 2-2.1 x
TSEI_10		Hijas De orden indiferente. Una repetitiva min=2, Max=4, la otra opcional	Secuencias correctas: (1-1.1)[2..4] (1-1.1)[2..4] – 2 2 - (1-1.1)[2..4]

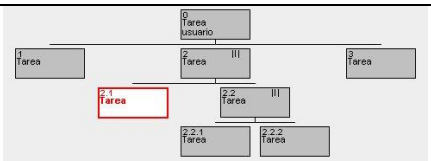
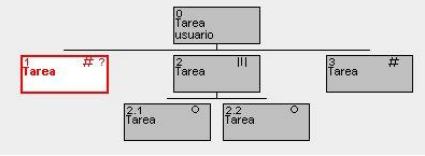
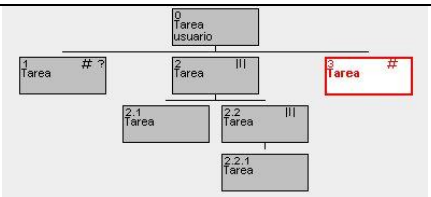
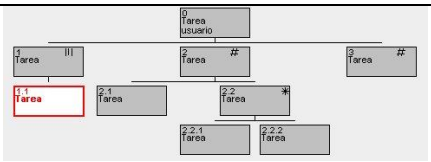
Nombre	Figura	Descripción	Efecto
TSEI_11		Hijas de orden indiferente. Una repetitiva min=1. Max=999, la otra opcional	Secuencias correctas: (1-1.1-1.2)[1..999] (1-1.1-1.2)[1..999] – 2 2 - (1-1.1-1.2)[1..999]
TSEI_12		Orden indiferente. Iterativa min=2 Max=999	Secuencias correctas: (1-1.1-1.2)*[2..999]-2 2-(1-1.1-1.2)*[2..999]
TSEI_13		Varias hijas, y nietas, de orden secuencial, elección e indiferente	Secuencias correctas: 1-2-2.1-3 1-2-2.2-2.2.1-2.2.2-3 1-2-2.2-2.2.2-2.2.1-3
TSEI_14		Varias hijas, y nietas, de orden secuencial, elección e indiferente	Secuencias correctas: 1-2-2.1-3 1-2-2.2-2.2.1-2.2.2-3 2-2.1-1-3 2-2.1-3-1 2-2.2-2.2.1-2.2.2-1-3 2-2.2-2.2.1-2.2.2-3-1 3-1-2-2.1 3-1-2-2.2-2.2.1-2.2.2

Nombre	Figura	Descripción	Efecto
TSEI_15		Varias hijas, y nietas, de orden secuencial, elección e indiferente. Incluye tipo repetitivo con orden indiferente.	Secuencias correctas: 1-(2-2.1|2-2.2-2.2.1-2.2.2)*-3 (2-2.1|2-2.2-2.2.1-2.2.2)*-1-3 (2-2.1|2-2.2-2.2.1-2.2.2)*-3-1 3-1-(2-2.1|2-2.2-2.2.1-2.2.2)* 3-(2-2.1|2-2.2-2.2.1-2.2.2)*-1
TSEI_16		Varias hijas, y nietas, de orden secuencial, elección e indiferente. Incluye tipo repetitivo con orden elección.	Secuencias correctas: 1-2-(2.1|(2.2-2.2.1-2.2.2)*)-3 2-(2.1|(2-2.2-2.2.1-2.2.2)*)-1-3 2-(2.1|2-2.2-2.2.1-2.2.2)*)-3-1 3-1-2-(2.1|(2-2.2-2.2.1-2.2.2)*) 3-2-(2.1|2-2.2-2.2.1-2.2.2)*)-1
TSEI_5		Varias hijas, y nietas, de orden secuencial, elección e indiferente, con diferentes tipos.	Secuencias correctas: x 1 2-(2.1-2.2-(2.2.1|2.2.2-2.2.2.1) 1-2-(2.1-2.2-(2.2.1|2.2.2-2.2.2.1)

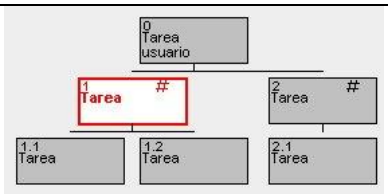
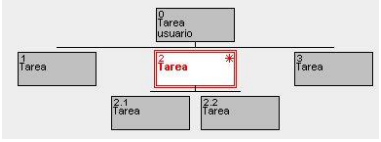
U.T. Objetivo 7-tSEIC: tSEI+Pruebas de integración: Tareas secuenciales-Tareas elección-Tareas indiferentes-tareas concurrentes en ventanas.

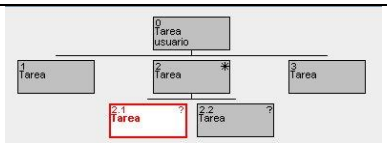
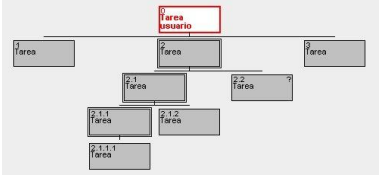
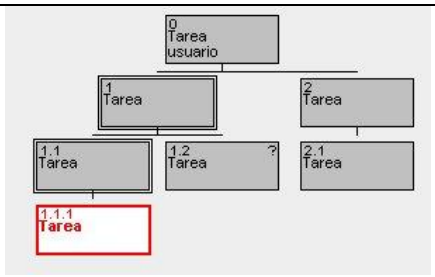
Nombre	Figura	Descripción	Efecto
TSEIC_1		Una tarea principal con una hija concurrente	Una ventana con un botón. Se pulsa, aparecerá una ventana de información que se cierra y vuelve al estado inicial. Finaliza correctamente si se cierra la ventana Secuencias: 1-x, x, (1)*-x
TSEIC_2		Una tarea principal con una hija concurrente y tipo opcional	Desactivado: Si es concurrente no será posible otro tipo u orden.
TSEIC_3		Varias hijas, una hija o varias son concurrentes sin hijas	En la ventana actual aparecerán siempre activas las tareas concurrentes, que podrán hacerse tantas veces como se quiera y en el orden que se quiera, o no activarlas ninguna vez. Finaliza correctamente cerrando la ventana, si las tareas obligatorias se han efectuado en el orden y número correcto. En este caso, solo hay una ventana activa.
TSEIC_4		Varias hijas, una hija o varias son concurrentes con hijas	En este caso, hay tantas ventanas activas como tareas concurrentes se estén efectuando. La tarea concurrente (sus hijas) se ejecuta en una ventana en paralelo con la ventana principal.

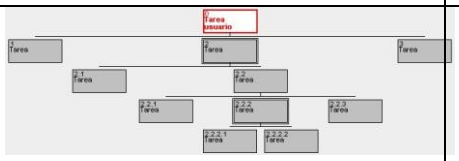
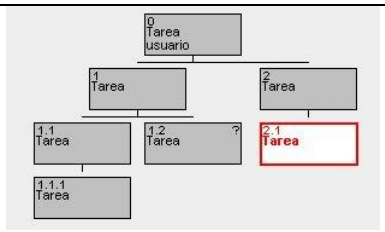
Nombre	Figura	Descripción	Efecto
TSEIC-4		Una hija concurrente hija de una tarea opcional	(1-1.1*)?-2
TSEIC-5		Una hija concurrente hija de una tarea repetitiva	En la ventana actual aparecerán siempre activas la tarea concurrente Secuencias correctas: (1-1.1*)*-2
TSEIC_6		Una tarea concurrente con hermanas de tipo elección	Aparece siempre activa la tarea concurrente. Se ha de cerrar la ventana. 3*-1-3* 3*-2-3*
TSEIC_8		Varias concurrentes en una ventana	Finaliza cerrando la ventana. (2*-3*)*-1-(2*-3*)*
TSEIC_9 ¿La concurrente puede interrumpir la iteración? SI		Concurrentes en varias ventanas. Repetitivas hermanas de concurrentes.	Las ventanas que tienen concurrentes, cuando finalizan las tareas obligatorias, deben cerrarse si no desea ejecutarse más veces las concurrentes situadas en esa ventana. **Ventana 1**: 2*-1-2*-3* (ventana2)-2* **Ventana 2**: 3.2*-3.1-3.2*

Nombre	Figura	Descripción	Efecto
TSEIC_10		Concurrentes con hijas en nueva ventana	Deben cerrarse **en orden** cuando se efectúan todas las tareas obligatorias de la ventana correspondiente. **Ventana 1**: 2*(ventana2)-1-2*(ventana2)-3-2*(ventana2) **Ventana 2**: 2.2*(ventana3)-2.1-2.2*(ventana3) **Ventana 3**: 2.2.1-2.2.2
TSEIC_11		Una tarea concurrente con hermanas de tipo indiferente	Sólo una ventana. 1-2*-3 3-2*-1 Debe cerrarse la ventana para concluir el proceso.
TSEIC_12		Una tarea concurrente con hermanas de tipo indiferente. La concurrente tiene hijas.	Habrá tantas ventanas en paralelo como activaciones de la tarea concurrente. 1?-(2-(2.1||2.2))*-3 3-(2-(2.1||2.2))*-1?
TSEIC_13		Una tarea concurrente con hermanas de tipo indiferente. La concurrente tiene hijas concurrentes	Deben cerrarse en orden inverso: V3-V2-V1 V 1: 1?-2*-3 V 2(en paralelo): 2.1-2.2 V 3(en paralelo): 2.2.1
TSEIC_14		Una tarea concurrente con hermanas de tipo indiferente.	V1:1*-(2-(2.1-2-2.2-2.2.1-2.2.2)*)-3||3-2-(2.1-2-2.2-2.2.1-2.2.2)*)) V2(en paralelo): 1.1

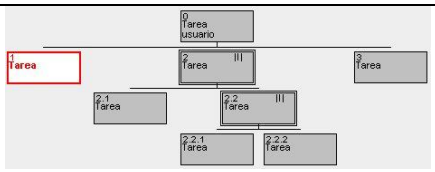
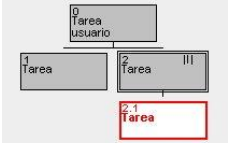
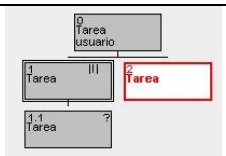
U.T. Objetivo 4-tSEIP: tSEI+ Pruebas del Modelo de Presentación

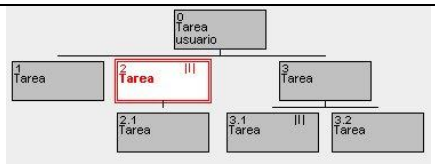
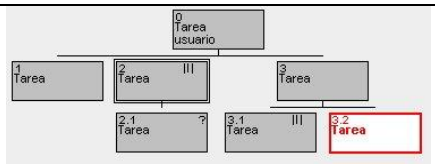
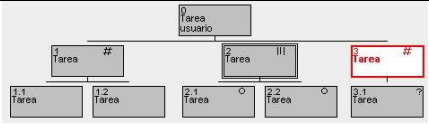
Nombre	Figura	Descripción	Efecto
TSEIP_0		Una tarea principal	**Desactivado**: Presentación. La tarea principal se mostrará siempre en una nueva ventana
TSEIP_1		Una tarea principal con una hija	**Desactivado**: Presentación. Una tarea sin hijas no tiene características de presentación para sus hijas.
TSEIP_2		Una tarea con hijas tiene modo de presentación “Nueva Ventana”	Ventana 1: 1 Ventana 2: 1.1- 1.2 Ventana 1: 2 Ventana 3: 2.1
TSEIP_3		Una tarea con hijas tiene modo de presentación “Nueva sección”	Ventana 1: 1-2 Ventana 1, foco en nueva sección: 2.1-2.2 (desaparece sección) Ventana 1: 3
TSEIP_4		Una tarea de orden indiferente y modo de presentación “Nueva sección”	Ventana 1: 1-2-(**foco en nueva sección**)2.1-2.2 -(desaparece sección)3-4, Combinaciones de orden:1-2-3-4
TSEIP_6		Una tarea secuencial, iterativa y modo de presentación “Nueva sección”	Ventana 1: 1-(2- **(foco en nueva sección**)2.1-2.2(desaparecen 2.1,2.2))*-3
TSEIP_7		Una tarea iterativa, con modo de presentación “Nueva sección”, con dos hijas opcionales	Ventana 1: 1-(2-2.1?-2.2?)*-3(desaparecen 2.1,2.2)

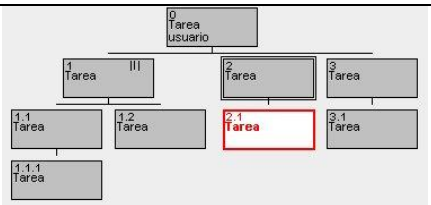
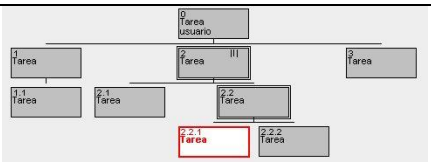
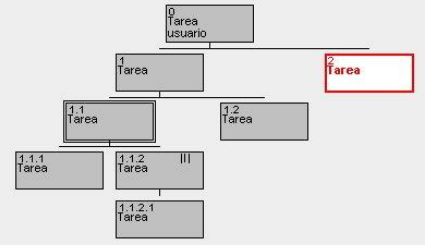
Nombre	Figura	Descripción	Efecto
TSEIP_8		Una tarea iterativa, con modo de presentación “Nueva ventana”, con dos hijas opcionales	Sec.: 1-(2-(2.1?-2.2?, x))*-3 Ventana 1: 1-2 Ventana 2: 2.1?-2.2?,X Ventana 1: 2 - 3
TSEIP_9 No habrá aviso , sino ajuste según tamaño de la resolución de la pantalla.		Enlaza tres “Nueva sección”.	Ventana 1, sección 1: 1-2 Sección 2: 2.1 Sección 3(aviso?): 2.1.1 Sección 4: 2.1.1.1 Sección 3: 2.1.2 Sección 2: 2.2? Sección 1: 3
TSEIP_10		Enlaza nuevas secciones, hermanas opcionales	V1: 1 S2:1.1 S3: 1.1.1 S2:1.2?|| S1:2 V1:2 V2:2.1

Nombre	Figura	Descripción	Efecto
TSEIP_11		Mezcla tipos de presentaciones.	El foco debe actualizarse en cada sección, y la activación de tareas debe señalar únicamente las posibles en cada momento. Ventana 1:1,2 Sección 1: 2.1,2.2 Ventana 2: 2.3.1,2.3.2 Sección 2: 2.3.1.1, 2.3.2.2 Ventana 2: 2.3.3 Ventana 1: 3
TSEIP_12		Con presentación en nueva ventana.	V1: 1 V2:1.1 V3: 1.1.1 V2:1.2 || X V1:2 V2:2.1

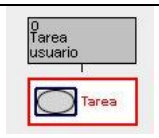
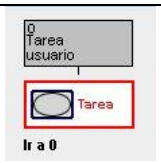
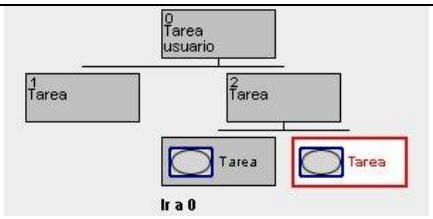
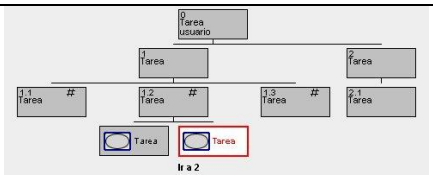
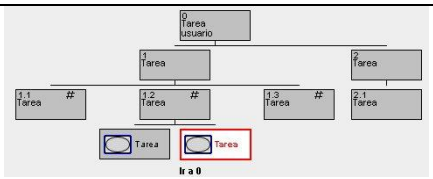
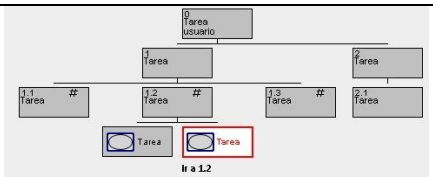
U.T. Objetivo 8: TSEIC + tSEIP + Pruebas de integración: Tareas secuenciales-Tareas elección-Tareas indiferentes-tareas concurrentes-Tareas de Presentación(nueva sección)

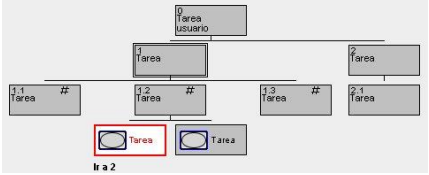
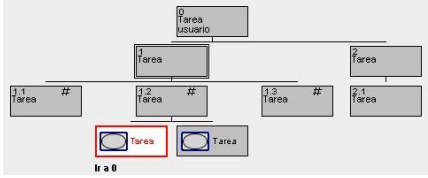
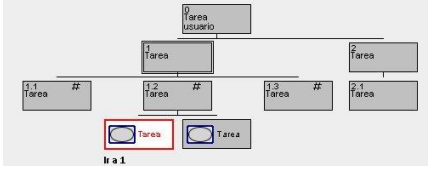
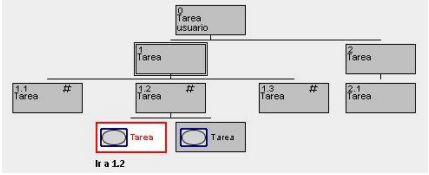
Nombre	Figura	Descripción	Efecto
TSEICP_1		Varias concurrentes encadenadas en secciones nuevas	Las secciones que tienen tareas concurrentes pendientes, únicamente desaparecen al pasar a otra activación de la sección anterior. **Ventana 1**: 2(sección2)*-1-2(sección2)*-3-2(sección2)* **Sección 2**: 2.2(sección 3)*-2.1-2.2 (sección 3)* **Sección 3**: 2.2.1-2.2.2
TSEICP_2		Concurrente con hijas en nueva sección	Las hijas de la tarea concurrente aparecerán en una nueva sección. V1: [2(S2)]*-1- [2(S2)]* S2: 2.1 Permanence activa siempre la concurrente 2
TSEICP_3		Concurrentes con hijas opcionales en nueva sección	V1,S1: 1(S.2)*-2-1(S.2)* Se deshabilita 1.1 al hacer la secuencia 1-2.

TSEICP_4		Concurrentes con hijas en nueva sección, con hermanas no concurrentes	La tarea 2 se ejecutará en una nueva sección. Sin embargo, al hacer la tarea 3 solo estará activa la ventana principal. **Ventana 1**: 2(sección2)*-1-2(sección2)*-3(Ventana 2)-2(sección2)* **Sección 2**: 2.1 **Ventana 1:**: 3.1*-3.2-3.1*
TSEICP_5		Concurrentes, con hijas opcionales en nueva sección, y con hermanas no concurrentes	La tarea 2 se ejecutará en una nueva sección. Sin embargo, al hacer la tarea 3 solo estará activa la ventana principal. **Ventana 1**: [2(sección2)]*-1-2(sección2)*-3(Ventana 2)-[2(sección2)]* **Sección 2**: 2.1? **Ventana 1:**: 3.1*-3.2-3.1*
TSEICP_6		Concurrentes, con hijas elección, y hermanas indiferentes.	La tarea 2 se ejecutará en una nueva sección. V1: [2(S2)]*-1-[2(S2)]*-3-[2(S2)]* || [2(S2)]*-3-[2(S2)]*-1-[2(S2)]* S2: 2.1|| 2.2
TSEICP_7		Concurrentes, con hijas elección, y hermanas indiferentes. Las tareas indiferentes tienen hijas secuenciales y opcionales	La tarea 2 se ejecutará en una nueva sección. La tarea 1 y la tarea 3 se ejecutarán en la ventana principal V1. V1: [2(S2)]*-1(V1')-[2(S2)]*-3(V1'')-[2(S2)]* || [2(S2)]*-3-[2(S2)]*-1-[2(S2)]* S2: 2.1|| 2.2 V1': 1.1-1.2 V1'':3.1-X

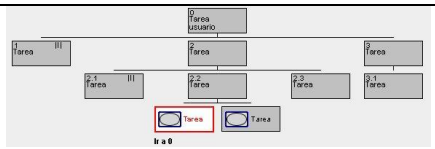
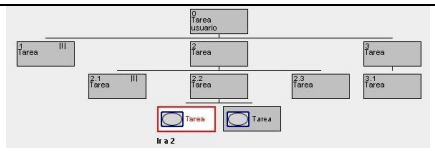
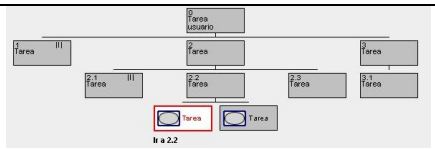
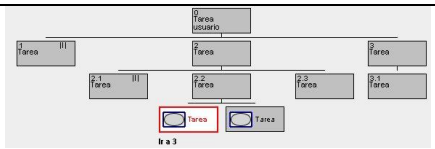
TSEICP_8		Tarea que se efectuará en nueva sección, con hermanas concurrentes	La tarea 2 se ejecutará en una nueva sección. En cualquier momento se puede activar la tarea 3. V1: 3*-1-3*-2(S2)-3* S2: 2.1-2.2
TSEICP_9		Tarea que se efectuará en nueva sección, con hermanas concurrentes que tienen hijas opcionales	La tarea 2 se ejecutará en una nueva sección. En cualquier momento se puede activar la tarea 1. Se puede interrumpir en cualquier momento la ejecución de la concurrente 1 para realizar el resto de las tareas. Una vez hecha la tarea 1.1, se puede cerrar la ventana si no se desea hacer la tarea 1.2
TSEICP_8		Tarea que se efectuará en nueva sección, con hermanas concurrentes que tienen hijas secuenciales	La tarea 2 se ejecutará en una nueva sección. En cualquier momento se puede activar la tarea 1. Se puede interrumpir en cualquier momento la ejecución de la concurrente 1 para realizar el resto de las tareas.
TSEICP_9		Una concurrente con una hija en nueva sección	La tarea 2 se ejecutará en una nueva sección en paralelo con la realización en la ventana principal sección 1.
TSEICP_10		Una tarea concurrente en una nueva sección, con hijas.	La tarea 1.1 se ejecutará en una nueva sección. La nueva sección desaparece cuando se han realizado todas las tareas obligatorias de la sección y se pulsa 1.2 en la sección anterior.4

Objetivo 6: tSEIP+ Integración (Secuenciales, elección, indiferentes) +Sistema + Modelo de Comportamiento +Presentación

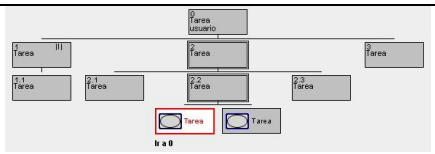
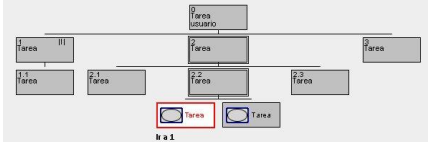
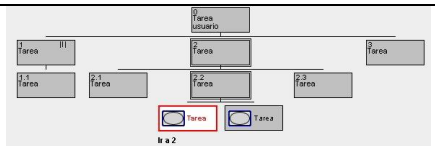
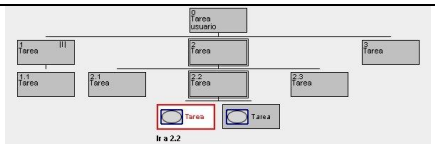
Nombre	Figura	Descripción	Efecto
TSEIBP_1		Tarea del Sistema	**Desactivado**: Orden , Tipo, Concurrente **Aviso**: No pueden tener hijos
TSEIBP 2		Tarea del Sistema con ir a..	• Sólo es posible en hojas • **Mensaje**: "Efectuando tarea X" • Se dirige a tareas objetivo(con hijos)
TSEIBP 3		Varias tareas del sistema	• Si una hermana es del Sistema, todas sus hermanas deben ser del Sistema. • Las tareas del Sistema sin etiqueta ir a.. van a la siguiente tarea a efectuar. Si es la última es "fin correcto"
TSEIBP_4		Tarea del Sistema, hija de indiferente, con etiqueta "ir a 2"	Borra las acciones pendientes, saca mensaje "*Efectuando tarea 2*", y comienza desde la tarea 2
TSEIBP 5		Tarea del Sistema, hija de indiferente, con etiqueta "ir a 0"	Borra las acciones pendientes(hacia abajo), saca mensaje "*Efectuando tarea 0*", y comienza desde la tarea 0
TSEIBP 6		Tarea del Sistema, hija de indiferente, con etiqueta "ir a 1.2"	Borra las acciones pendientes, saca mensaje "*Efectuando tarea 1.2*", y comienza desde la tarea 1.2. Tendrá en cuenta el contexto (orden/tipo/concurrencia) de la tarea 1.2

TSEIBP_7		Nueva sección Tarea del Sistema, hija de indiferente, con etiqueta “ir a 2”	Borra las acciones pendientes y la ventana con nueva sección, saca mensaje “*Efectuando tarea 2*”, y comienza desde la tarea 2 en una nueva ventana.
TSEIBP_8		Nueva sección. Tarea del Sistema, hija de indiferente, con etiqueta “ir a 0”	Borra las acciones pendientes , y la ventana con nueva sección, saca el mensaje “*Efectuando tarea 0*”, y comienza desde la tarea 0 en una nueva ventana.
TSEIBP_9		Nueva sección. Tarea del Sistema, hija de indiferente, con etiqueta “ir a 1”	Borra las acciones pendientes, saca mensaje “*Efectuando tarea 1*”, y comienza desde la tarea 1, en una nueva sección.
TSEIBP_10		Nueva sección. Tarea del Sistema, hija de indiferente, con etiqueta “ir a 1.2”	Borra la última ventana, saca mensaje “*Efectuando tarea 1.2*”, y comienza desde la tarea 1.2, en una nueva ventana. Tiene en cuenta el contexto de la tarea1.2 para saber cuál es la siguiente tarea.

Objetivo 9: tSEIB+ TSEIC+ Integración (Secuenciales, elección, indiferentes) +Sistema + Comportamiento +Concurrentes

Nombre	Figura	Descripción	Efecto
TSEIBC_1		Tareas de distinto orden y concurrentes Tarea del Sistema, con etiqueta “ir a 0”	Borra las acciones (y ventanas) pendientes , saca mensaje “*Efectuando tarea0*”, y comienza desde la tarea 0 en una nueva ventana.
TSEIBC_2		Tareas de distinto orden y concurrentes Tarea del Sistema, con etiqueta “ir a 2”	Borra acciones y ventanas pendientes desde la tarea 2. Saca el mensaje “*Efectuando tarea 2*”, y comienza desde la tarea 2. Tendrá en cuenta el contexto de la tarea 2.
TSEIBC_3		Tareas de distinto orden y concurrentes Tarea del Sistema, con etiqueta “ir a 2.2”	Borra acciones y ventanas pendientes desde la tarea 2. Saca el mensaje “*Efectuando tarea 2.2*”, y comienza desde la tarea 2.2. Tendrá en cuenta el contexto de la tarea 2.2.
TSEIBC_4		Tareas de distinto orden y concurrentes Tarea del Sistema, con etiqueta “ir a 3”	Borra las acciones (y ventanas) pendientes , saca mensaje “*Efectuando tarea 3*”, y comienza desde la tarea 3 en una nueva ventana

Objetivo 10: tSEIBC+ TSEIBP+ Integración (Secuenciales, elección, indiferentes) +Sistema + Comportamiento +Concurrentes+Presentación

Nombre	Figura	Descripción	Efecto
TSEIBCP_1		Tareas concurrentes Tarea del Sistema, con etiqueta “ir a 0”en nueva sección	Borra las acciones (y ventanas) pendientes , saca mensaje “*Efectuando tarea0*”, y comienza desde la tarea 0 en una nueva ventana. Las concurrentes previamente activadas desaparecen.
TSEIBCP_2		Tareas concurrentes Tarea del Sistema, con etiqueta “ir a 1”en nueva sección	Borra acciones y ventanas pendientes desde la tarea 2. Saca el mensaje “*Efectuando tarea 1*” en una nueva ventana concurrente (verde). Las ventanas concurrentes anteriores se mantienen.
TSEIBC_3		Tareas concurrentes Tarea del Sistema, con etiqueta “ir a 2”en nueva sección	Borra acciones y secciones pendientes desde la tarea 2. Saca el mensaje “*Efectuando tarea 2*”, y comienza desde la tarea 2 en una nueva sección. Tendrá en cuenta el contexto de la tarea 2.
TSEIBC_4		Tareas concurrentes Tarea del Sistema, con etiqueta “ir a 2.2”en nueva sección	Borra las acciones (y ventanas) pendientes , saca mensaje “*Efectuando tarea 2.2*”, y comienza desde la tarea 2.2 en una nueva sección

Visualización en el diagrama:

Las encuadramos en estos cuatro tipos: una tarea inicial, una tarea hija sin hijas, una tarea con hija con hijas y una tarea concurrente.

T0.tareas: Una tarea inicial

En el diagrama:

OPCIONES	EJECUTABLE	MODO
Nombre, informe, comentario	S	
Crear hijas hacia abajo	S	
Crear tarea a la izda	N	**Aviso:** En la tarea ppal solo se pueden crear hijas hacia abajo
Crear hija a la dcha	N	**Aviso:** En la tarea ppal solo se pueden crear hijas hacia abajo
Mover tarea a la izda	N	**Aviso:** La tarea ppal no se puede mover a la izda
Mover tarea a la dcha	N	**Aviso:** La tarea ppal no se puede mover a la dcha
Borrar	N	**Aviso:** La tarea ppal no se puede borrar
Orden	N	**Desactivado**
Tipo	N	**Desactivado**
Presentación	N	**Desactivado**
Ir a	N	**Desactivado**
Conversión U/S	N	**Aviso:** La tarea ppal no puede ser una tarea del Sistema.
Glosar/Desglosar	N	**Aviso:** La tarea ppal no se puede desglosar

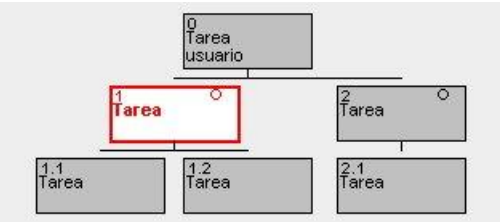
T1: una tarea hija sin hijas

Para la tarea 0, las opciones activas y no activas son las especificadas en la primera prueba.

Tarea 1:

OPCIONES	EJECUTABLE	MODO
Nombre, informe, comentario	S	
Crear hijas hacia abajo	S	
Crear tarea a la izda	S	
Crear hija a la dcha	S	
Mover tarea a la izda	N	**Aviso**: No hay más tareas a la izda
Mover tarea a la dcha	N	**Aviso**: No hay más tareas a la dcha
Borrar	S	
Orden	S	**Aviso**: No es posible orden elección-tipo opcional
Tipo	S	**Aviso**: No es posible un tarea hoja repetitiva. **Aviso**: No es posible orden elección-tipo opcional
Presentación	N	**Desactivado** en todas las tareas hojas
Ir a	S	
Conversión U/S	S	
Glosar/Desglosar	S	

T2: una tarea con hija con hijas



Para la tarea 0, las opciones activas y no activas son las especificadas en la primera prueba.

Tarea 1:

OPCIONES	EJECUTABLE	MODO
Nombre, informe, comentario	S	
Crear hijas hacia abajo	S	
Crear tarea a la izda	S	
Crear hija a la dcha	S	
Mover tarea a la izda	N	**Aviso**: No hay más tareas a la izda
Mover tarea a la dcha	S	
Borrar	S	**Aviso**: Las subtareas de esta tarea serán eliminarlas. ¿Desea borrar la tarea?
Orden	S	
Tipo	S	**Aviso**: No es posible orden elección-tipo opcional **Aviso**: Repetitiva Min>Max o Min=Max=0 **Aviso**: No es posible orden elección–tipo concurrente **Aviso**: No es posible orden indiferente-tipo concurrente
Presentación	S	
Ir a	N	**Desactivado** en todos los nodos intermedios

OPCIONES	EJECUTABLE	MODO
Conversión U/S	N	**Aviso**: Una tarea con hijos no puede ser del sistema.
Glosar/Desglosar	S	**Aviso**: Al desglosar

TC_1: Una tarea concurrente

Para la tarea 0, las opciones activas y no activas son las especificadas en la primera prueba.

Tarea 1:

OPCIONES	EJECUTABLE	MODO
Nombre, informe, comentario	S	
Crear hijas hacia abajo	S	
Crear tarea a la izda	S	
Crear hija a la dcha	S	
Mover tarea a la izda	N	**Aviso**: No hay más tareas a la izda.
Mover tarea a la dcha	N	**Aviso**: No hay más tareas a la dcha.
Borrar	S	
Orden	N	**Desactivado:** Las tareas concurrentes no son de ningún orden.
Tipo	N	**Desactivado:** Las tareas concurrentes no son de ningún tipo.
Presentación	N	**Desactivado** en todas las tareas hojas
Ir a	S	

OPCIONES	EJECUTABLE	MODO
Conversión U/S	S	Pierde la característica de concurrencia
Glosar/Desglosar	S	**Aviso**: Al desglosar