TRABAJO DE GRADO

Herramienta asistente para el análisis de requerimientos

Director

Doctor Gustavo Rossi

Codirector

Licenciado Federico Balaguer

Alumno

Rubén Leandro Antonelli

Organización del trabajo

El presente documento se haya estructurado de la siguiente forma. Esta dividido en dos grandes partes. En la primera se expone toda la teoría analizada para la construcción de la herramienta. En la segunda se documenta la herramienta. A continuación se detallan los contenidos del trabajo.

1.Teoría

Análisis: 1. separación de cualquier material o entidad abstracta en sus elementos constituyentes. 2. método de estudio de la naturaleza de algo, determinación de sus rasgos esenciales y sus relaciones. (Random House Dictionary of the English Languaje)

1.1.Introducción

El desarrollo de cualquier sistema de soft implica el entendimiento y abstracción del dominio, es decir de los elementos del dominio, y de la lógica y reglas que gobiernan su comportamiento. Una aproximación orientada a objetos requiere que este conocimiento sea modelado como un conjunto de objetos en colaboración. Por esta razón, se necesitan herramientas y técnicas que permitan la representación correcta del dominio y su traslado hacia un modelo de objetos.

1.1.1.Motivación

El desarrollo de soft es una tarea crítica. Desde que el usuario pide la construcción de una aplicación, hasta el último día que se la utilice, el desarrollador se enfrenta a varios problemas. En primer lugar debe entender exactamente que es lo que el cliente necesita. Luego debe construir una solución modificable y reusable. Y durante toda la vida del soft adaptarlo a las especificaciones oportunas del usuario.

Con el fin de servir como guía y agilizar el proceso desarrollo de soft, se concluyeron varias metodologías. Los modelos mas populares que llegan hasta nuestros días son: modelo en cascada y prototipación.

El modelo en cascada consiste en dividir la construcción de soft en etapas: definición, diseño, implementación, prueba y mantenimiento. Básicamente la idea de cada etapa es la siguiente. En la fase de definición se analiza que se desea hacer, se estudia la viabilidad de la construcción del soft desde diferentes puntos de vista, y se intenta obtener una especificación detallada del comportamiento del soft a implementar. Esto último se conoce como análisis de requerimientos y es, tal vez, la parte mas conflictiva. Por lo general es muy dificil que un usuario presente una especificación completa y sin ambigüedades del soft que desea. La segunda etapa es la del diseño. En ella se hace un bosquejo del soft por medio de técnicas como el diseño estructurado, orientas al flujo de información y a los objetos. Este diseño se utilizará como base para la implementación en un lenguaje concreto en la siguiente etapa. Al ya tener la aplicación ejecutable en máquina, se comienza con la siguiente etapa que es la de prueba. Se puede considerar a esta etapa como una fase destructiva en la cual se somete al soft a toda clase de prueba para detectar si tiene alguna falla. Por último se llega a la etapa de mantenimiento. Esta etapa es la que se realiza desde que se instala el soft y se lo empieza a utilizar. El mantenimiento puede ser de tres tipos: adaptativo, evolutivo y correctivo. Que consiste en corregir errores que no se detectaron en la fase de prueba o ampliar el soft a medida que la realidad del sistema evoluciona. Las etapas son secuenciales y no se pueden superponer. Por ejemplo antes de diseñar es necesario concluir la definición. Estas cinco etapas no son absolutas, algunos autores incluyen la instalación como una etapa adicional. Las etapas se dividen a su vez en subetapas. Dentro de definición se realizan las tareas de definición del problema, análisis de viabilidad, análisis de requerimientos, documentación, entre otras. Al igual que las etapas, distintos autores tienen distintos puntos de vista sobre las subetapas.

El segundo modelo utilizado en la actualidad es el de prototipación. El modelo en cascada tiene el problema de que el usuario no ve el producto final hasta que termine el desarrollo. El modelo de prototipación se basa en un modelo en espiral, en el cual se van sucediendo las etapas del modelo en cascada. Se podría resumir la idea de la siguiente forma: lo que se hace es desarrollar por tramos, a medida que se recolectan requerimientos. De esta forma, el usuario está en contacto con el ingeniero de soft durante todo el desarrollo del soft, para transmitir a este los requerimientos e ir probando la aplicación. Como no espera hasta ver el producto final, se pueden detectar errores prematuramente, cuya corrección no requiera tanto esfuerzo.

Si bien ambos modelos ofrecen una buena ayuda para el desarrollo de soft, tienen el inconveniente de que dicen "que" es lo que hay que hacer, pero sin decir el "como". Este trabajo tiene por objetivo plantear el "como" para la fase de análisis de requisitos y diseño preliminar, en un ambiente de objetos. Se refinan métodos y heurísticas del modelo del baseline tratado en [Leite 90], [Leite 93], [Leite 95b] y [Leite 97c]. Y se implementa una herramienta que soporta el modelo presentado.

La organización del trabajo es la siguiente: en los puntos 1.1.2, 1.1.3 y 1.1.4 se da una idea introductoria y global de todo el modelo. En los puntos 1.2, 1.3 y 1.4 se tratan en profundidad cada uno de los pilares del modelo: LEL, Escenario y clases candidatas, respectivamente. Y por último, en la sección 2 se trata la herramienta que automatiza el modelo presentado.

1.1.2.Baseline

Viendo la necesidad de brindar métodos y herramientas para sistematizar el proceso de análisis de requerimientos, en [Leite 90] se propone **A client oriented requirements baseline**, que está compuesta de la siguiente forma.

1.1.2.1.Basic Model

En primer lugar se tiene una estructura denominada **basic model**, que incorpora oraciones sobre el sistema deseado. Estas oraciones se escriben en lenguaje natural siguiendo formatos preestablecidos. Las mismas son escritas por el ingeniero de soft pero son validadas por el cliente. Tanto esta herramienta como las que se ven a continuación no son estáticas. Se desarrollan durante el proceso de ingeniería de requerimientos pero siguen evolucionando a medida que el soft es construido. Con respecto a esta primera herramienta tratada en [Leite 90], los trabajos que los suceden tomas dos caminos distintos. Están aquellos que consideran al basic una ayuda y los siguen utilizando; y también los que piensan que no es nada simple escribir estas oraciones obedeciendo al formato preestablecido y deciden prescindir de ella. En este trabajo se va a prescindir de este basic model. En la Ilustración 1 se puede ver el diagrama entidad relación que describe cual es la estructura que deberían tener las oraciones del basic model y a continuación la descripción del mismo, pero no se lo sigue tratando a lo largo del trabajo.

- Client: Es una persona o grupo responsable de una acción. Tiene la estructura siguiente:

  Nombre + Ubicación

- Action: Es una acción concreta realizada por Client en su trabajo. Action es representada por una oración con la siguiente estructura:

  Verbo en infinitivo + Predicado + [Identificador Action Padre] + { Atributos restrictivos } + { Atributos diagnósticos }

  Una action se puede descomponer en sub-actions con la misma estructura de una action.

- External Event: Ocurre en el universo del discurso y debería originar una respuesta del sistema. Cada evento se debe asociar a una Action existente.

  [Identificador Temporal Stimuli | Identificador External Stimuli] + Identificador Action + [External Event Precedentes] + { Atributos restrictivos } + { Atributos diagnósticos }

- Temporal Stimuli: Instante de tiempo en el cual el sistema tendrá que reaccionar independientemente del input

  Referencia temporal + Sujeto + Verbo + Predicado

- External Stimuli: Un acontecimiento en el universo del discurso que requiere la reacción del sistema:

  Sujeto + Verbo + Predicado

- Output: Información que será producida por el sistema.

  Nombre + [Descripción] + { Atributos restrictivos } + { Atributos diagnósticos}

- Input: Información que es consumida por el sistema.

  Nombre + [Descripción] + { Atributos restrictivos } + { Atributos diagnósticos}

Los atributos mas importantes del modelo entidad relación son las restricciones y diagnósticos. Una restricción se refiere a una entidad y se representa por una oración breve con la estructura: debe + verbo + predicato. Un diagnóstico es una observación amplia un problema relacionado con una entidad. Los diagnósticos son los puntos centrales de la validacion de clientes.

1.1.2.2.Léxico extendido del lenguaje

El segundo elemento es el **léxico extendido del lenguaje** (LEL Languaje Extended Lexicon). Este es un modelo que captura los símbolos relevantes del vocabulario comúnmente usado por la gente que trabaja dentro del dominio de la aplicación. Cada símbolo es relacionado a sus sinónimos, a un conjunto de nociones (notions) y a un conjunto de impactos (behavioral responses). Las nociones son un conjunto de expresiones que definen a un símbolo. Y los impactos muestran la relación entre el término y el dominio, indicando como el símbolo afecta otros símbolos. Las nociones e impactos contienen links a otros símbolos, así un ambiente navegacional es construido. El LEL es gobernado por dos principios: el de mínimo vocabulario y el de circularidad. El principio de circularidad establece que hay que utilizar dentro de las nociones e impactos símbolos definidos en el mismo LEL. El de mínimo vocabulario, por su parte establece que los símbolos no pertenecientes al LEL que se utilicen deben poseer un connotación matemática (inclusión, diferencia, pertenencia).

1.1.2.3.Escenarios

La tercera componente de la baseline lo forman los escenarios. Estos son utilizados para modelar aspectos dinámicos del dominio. Los escenarios son descripciones parciales del comportamiento del sistema y del ambiente. Ellos tienen una

relación muy estrecha con la forma en que la gente entiende y describe problemas. Se utilizan escenarios para describir el macrosistema en el cual el sistema de software es insertado. Los escenarios se describen por: nombre (name), objetivo (goal), recursos (resources), contexto (context) y episodios (episodes).

Estos tres modelos, de los cuales se utilizan los dos últimos, brindan soporte al ingeniero de soft en la tarea de análisis de requerimientos. El objetivo de este trabajo es implementar una herramienta que integre estos dos modelos, abierta con la expectativa de que soporte próximas etapas del desarrollo de soft. Con este fin se le incorpora la funcionalidad necesaria para que realice los primeros pasos del diseño preliminar de objetos.

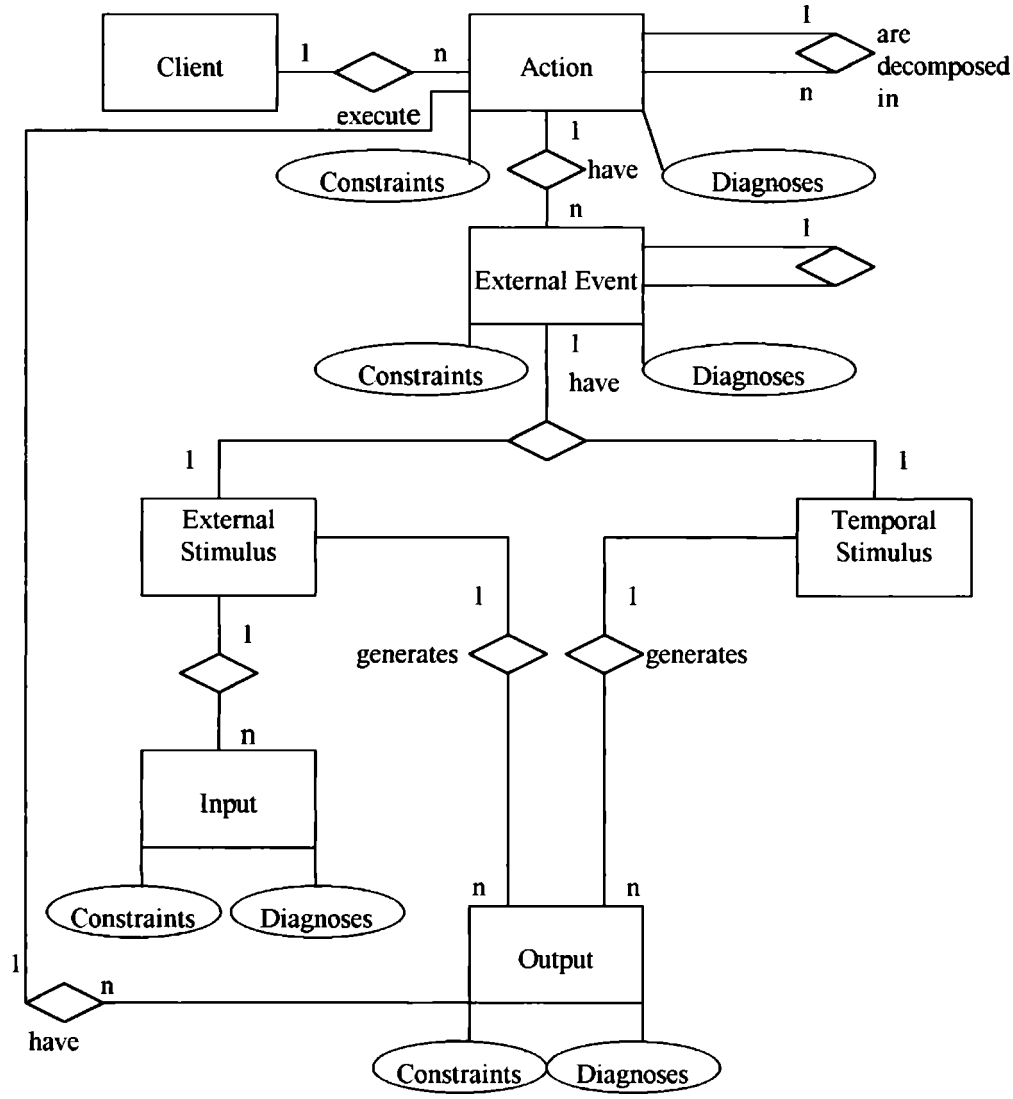


Ilustración 1. Diagrama entidad relación del formato de las oraciones del basic model

Lo que se hace es encontrar clases candidatas para construir el modelo preliminar de objetos. No hay que olvidar que la intención de este trabajo es desarrollar una herramienta para el desarrollo de aplicaciones en un ambiente de objetos. Así que con la ayuda del LEL y los escenarios se derivaran las clases candidatas. Las mismas tienen que ser analizadas con el usuario para determinar que esta dentro de los límites del soft y que no.

1.1.3.Análisis de requerimientos

En este apartado se amplian aspectos del LEL y escenarios. Sin embargo, una explicación completa y minuciosa se da en las secciones 1.2 y 1.3.

La recolección de requerimientos es un proceso en el cual el ingeniero de software trabaja para entender y descubrir que se espera de un sistema de software planeado. Aunque es una realidad que esta es una tarea cansadora, los investigadores atacan el problema utilizando diferentes **frameworks**. En ellos se trabaja con representaciones artificiales o lenguajes naturales. Este trabajo esta basado en el uso del lenguaje natural. Una razón importante para esta elección es evitar la introducción de lenguajes artificiales al comienzo del proceso de desarrollo de soft. La formalización debe ser un proceso gradual para tratar la gran brecha entre la realidad y el modelo formal.

El primer aspecto importante sobre el LEL está expuesto en el párrafo anterior. Esta orientado al uso de lenguaje natural. El segundo elemento no menos importante es el objetivo. Con el LEL no se pretende encontrar las funciones que el soft debe implementar. El objetivo es mas básico. Lo que él debe permitir es entender el lenguaje del universo del discurso. El lema que sustenta el LEL es:

"Entender el lenguaje del problema, sin preocuparse por entender el problema."

Además de apoyar la fase de ingeniería de requerimientos, el LEL es una ayuda para los procesos de reingeniería. En base al texto que describe una aplicación, sus diagramas de flujo y diccionario de datos, se puede capturar el universo de discurso de la aplicación, para luego derivar en la especificación del mismo.

El Léxico Extendido del Lenguaje intenta capturar el vocabulario de una aplicación. El fin principal del LEL es registrar signos (palabras o frases), las cuales son peculiares al dominio. Cada entrada en el LEL tiene dos tipos de descripciones, en contraposición con el diccionario usual que tiene sólo una. El primer tipo, denominado noción, es el tipo usual y su objetivo es describir lo que denota la palabra o frase. El segundo tipo, denominado impacto, tiene como objetivo describir información extra sobre el contexto. La idea del LEL proviene de la observación de que existe una fuerte relación entre cultura y lenguaje, tanto, como para que un universo de discurso dado este contenido dentro de un lenguaje propio. Una aproximación para comprender el lenguaje es listar sus signos y los significados. LEL, el cual, corresponde a esta idea, tiene tres entidades diferentes: signos, nociones e impactos.

Los signos (palabras o frases) son representaciones sintácticas obtenidas de una etapa de identificación de signos. Las nociones son los significados que podrían ser relacionados a un signo, describiéndolo desde diferentes puntos de vista. Los impactos son el efecto de un signo dentro del universo de discurso. Sobre esta representación básica, se establecen dos principios fundamentales:

- Cuando se describe una noción o un impacto, se debe **maximizar** el uso de signos del LEL. Se llama a este, *principio de circularidad*.

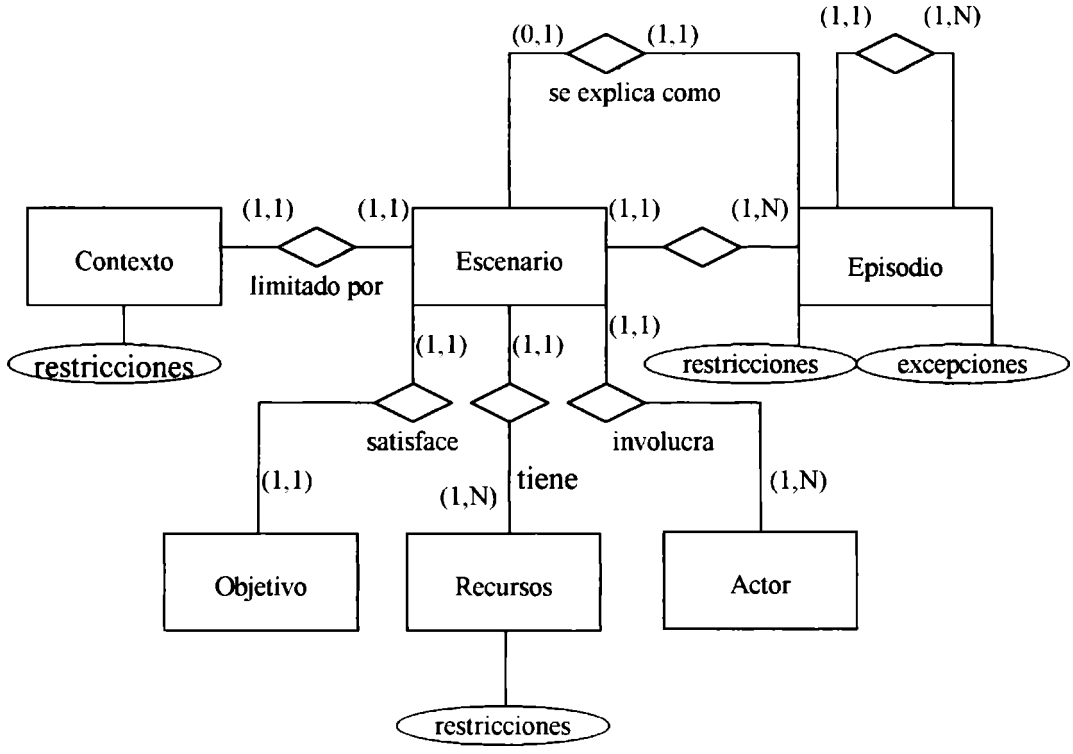


Ilustración 2. Diagrama entidad relación de la estructura de los escenarios.

- Cuando se describe una noción o un impacto, se debe **minimizar** el uso de signos externos al dominio. Cuando se lo hace se debe asegurar de que ellos pertenezcan a un vocabulario básico del lenguaje natural en uso y, tanto como sea posible, que tenga una representación matemática clara (por ejemplo: conjunto, pertenencia, intersección y función). Este principio es llamado, *principio de vocabulario mínimo*.

Imponiendo el principio de vocabulario mínimo y el de circularidad, se está formando un conjunto de signos que se definen en función de ellos mismos. Cada uno puede verse como un nodo de información, y la relación entre ellos como **links**. De esta forma, se está originando un grafo. Este grafo de texto, no es otra cosa que un hypertexto.

El segundo elemento que se utiliza para apoyar la etapa de análisis de requerimientos, es el modelo de escenarios.

Para entender la concepción de los escenarios, que se asume en este trabajo, se presentan cuatro premisas fundamentales.

- Un escenario describe situaciones, con énfasis en la descripción del comportamiento.
- Un escenario evoluciona a medida que se progresa en el proceso de construcción de soft.
- Los escenarios son naturalmente combinados al LEL.

- Los escenarios utilizan descripción textual como representación básica.

El modelo de escenarios es una estructura compuesta de objetivo (goal), contexto (context), recursos (resources), actores (actors) y episodios (episodes). Objetivo, contexto, recursos y actores, son oraciones declarativas. Episodios es un conjunto de oraciones en concordancia a un lenguaje muy simple que hace posible la descripción operacional del comportamiento.

Se describe al escenario usando el modelo entidad relación (Ilustración 2). Notar que, un episodio puede ser expresado como un escenario en si mismo, permitiendo dividir escenarios en subescenarios.

A continuación se describen las entidades pertenecientes al modelo ER.

- Titulo: El título del escenario. En el caso de un subescenario el título es el mismo que la oración del episodio.
- Objetivo: Es el fin que debe ser logrado.
- Contexto: Describe el estado inicial del escenario.
- Recursos: Es el soporte del que se dispondrá en el escenario.
- Actor: Es una persona o entidad que juega un rol dentro del escenario.
- Episodios: Una serie de oraciones que detallan el comportamiento del escenario.

De la misma forma que el LEL tiene una representación hypertextual, los escenarios también la tienen. Hay muchas relaciones a ser exploradas entre los escenarios y, entre escenarios y otros elementos del baseline (LEL). De cada escenario se derivan uno o mas nodos que son relacionados por links hypertextuales. Un link puede ser derivado de tres maneras diferentes:

- De una relación estructural existente entre escenarios. Por ejemplo la relación subescenario.
- De información contenida en el LEL. Las palabras o frases utilizadas para definir a los escenarios, que están definidas en el LEL, se ligan a su definición.
- Definida oportunamente. Podría quererse relacionar dos escenarios que no se relacionan estructuralmente o que tampoco comparten ningún ítem del LEL. Esta clase de link no puede ser generado automáticamente por la herramienta y debe ser explícitamente creado.

Otro aspecto en común que tendrán el LEL y los escenarios, es la posibilidad de manejar configuraciones. Una configuración es el estado del LEL y escenario en un instante de tiempo. De esta forma, el manejo de configuraciones permite hacer un seguimiento de toda la evolución de estos dos modelos. Ni el LEL, ni los escenarios son estáticos. Durante el desarrollo del soft, se van clarificando los requerimientos y se agregan nuevos elementos o se pueden modificar los ya existentes, a estos dos modelos. Para identificar con que versión del LEL y escenarios se corresponde el soft en desarrollo, es que se agrega este aspecto de las configuraciones.

1.1.4.Diseño preliminar

Se analiza una estrategia para construir un modelo preliminar de objetos combinando el conocimiento complementario dado por el LEL y los escenarios. Esta estrategia esta basada en varias heurísticas destinadas a identificar clases candidatas, sus responsabilidades y colaboraciones.

Como se menciona en la seccion 1.1.2, la idea de este trabajo no es desarrollar completamente "como" hacer el diseño en objetos, de modo que se hace la parte de diseño preliminar como para dejar la herramienta abierta a futuras ampliaciones.

En lo que respecta al LEL, cada entrada tiene noción e impacto. Si se analiza desde el punto de vista del diseño de objetos, al impacto se lo puede ver como las responsabilidades que el objeto tiene. De modo que hallar clases candidatas implica ubicar signos, que describan un impacto relevante dentro del dominio de la aplicación.

Una vez que se obtienen las clases primarias, se deben identificar las clases secundarias. Estas representan el recurso a las clases primarias para completar su responsabilidad. Así que se las tienen que buscar en los recursos del escenario que tiene como actor a la clase primaria y obtener sus responsabilidades del LEL.

Lo que se describe en los parrafos anteriores es una idea básica de como derivar un modelo preliminar de objetos de la información contenida por el LEL y escenarios. Con respecto a los escenarios, este es un modelo ya utilizado para análisis de requerimientos, muchos estudios se han realizado para traducir su información a un modelo de objetos. Sin embargo LEL es un modelo nuevo, que se lo presenta, con el fin de conocer el lenguaje del dominio, y como se puede apreciar, esta información es de gran ayuda para un modelo de objetos.

1.2. LEL

Para que la construccion de soft no sea un proceso artesanal, el LEL aporta la informacion del dominio de la aplicación, de forma que la especificación evolucione hasta convertirse en el modelo de objetos.

1.2.1. Introducción

La estrategia propuesta en este trabajo procura abordar la etapa de análisis de requerimientos centrándose en el aspecto del lenguaje de la aplicación.

El lenguaje de la aplicación es el lenguaje usado por los actores (personas que participan del ambiente de la aplicación) día a día y es una extensión del lenguaje natural usado normalmente. Una forma de obtener el lenguaje de aplicación es capturar los términos propios del mismo. A falta de un método mas preciso para hacer frente con el estudio del problema (mundo real) directamente, se torna el estudio del lenguaje de aplicación, una salida atrayente. Antes de hacer algo, es preciso conocer y entender que debe ser hecho.

Un lenguaje es la mayor expresión de una determinada cultura y a su vez es determinada por un medio social. Por lo tanto para una determinada aplicación existe un medio social con varios actores que, amoldan de cierta manera una cultura propia.

Este lenguaje puede contener palabras y construcciones específicas al ambiente de la aplicación, o bien expresiones del lenguaje natural con un significado distinto del corriente. Lo que se busca con el LEL es identificar estos símbolos, para su posterior definición con ayuda de los usuarios. Luego, el lema detrás del LEL es:

"entender el lenguaje del problema, sin preocuparse en entender el problema"

Desde el punto de vista de la construcción de los escenarios, el LEL es un tamiz del universo de discurso. Los signos identificados, forman la materia prima para los escenarios.

1.2.2.Estructura

En este punto se profundiza sobre la estructura del LEL. Para fijar más los conceptos se utiliza como ejemplo la siguiente especificación de requisitos de un sistema de ahorro:

> *El sistema se encarga de recaudar las cuotas de los titulares de un plan durante un cierto período, acreditándole intereses, luego de ese período se le adjudica un préstamo que está en función de lo abonado. Las cuotas que eran de ahorro pasan a ser de cancelación de deuda.*

Se recordará que el fin principal del LEL es registrar signos (palabras o frases), las cuales son peculiares al dominio. Así que lo que se debe hacer es tomar aquellas palabras o frases relevantes o aquellas cuyo significado es desconocido o parecieran estar fuera de contexto. La primera palabra que se encuentra es *titular*. Ya es posible armar una entrada en el LEL con el signo titular.

Signo: **titular**
Nocion:
Impacto:

Ilustración 3

Como se puede ver en la ficha, a cada signo se lo describe de dos forma. Por un lado con una definición al estilo del diccionario tradicional. A esta descripción, se la llama noción. Por otro lado, también se describe al signo en función del impacto. Esto es como influye el elemento que se describe en los demás elementos del universo del discurso y que incidencia tienen los demás elementos del discurso sobre el signo que se define. Ya que la especificación es muy escueta, para describir al signo en este ejemplo se tiene que recurrir a los usuarios. Los mismos definen a titular de la siguiente forma:

Signo: **titular**
Nocion: 1. Persona que es beneficiario directo de un plan de ahorro y préstamo. 2. Persona que paga las cuotas de un plan de ahorro. 3. Es quien recibe el monto del préstamo. 4. Es quien puede rescindir.
Impacto: 1. Realiza las acciones: solicitar admisión, pagar las cuotas, recibir el monto, rescindir.

Ilustración 4

A partir de esta ficha para el signo titular, se introduce el concepto de sinónimo. La idea de un sinónimo es la misma que en el lenguaje natural. En el LEL dos sinónimos son dos signos que representan el mismo elemento. Estos sinónimos que se pueden ver en el ejemplo son plan y, plan de ahorro y préstamo. En la especificación se menciona el término plan, y en la noción número uno de titular se menciona plan de ahorro y préstamo. Cuando se tengan sinónimos la forma de anotarlos es en la entrada del signo separados por barra:

Signo: **plan / plan de ahorro y préstamo**
Nocion:
Impacto:

Ilustración 5

Si se mira la definición de titular, allí se puede ver otro elemento importante. Esto es, para definir titular se utiliza en la noción número uno el término plan de ahorro y préstamo. Ahora es necesario definir este término. Esto no es casualidad, sino que responde a uno de los principio que rigen la construcción del LEL.

El primero es el principio de circularidad, que asevera que en la descripción de la noción y el impacto se debe maximizar el uso de signos del LEL. El segundo principio es el de vocabulario mínimo, el cual establece que hay que minimizar el uso de signos externos al domino. Cuando se los usa, se debe asegurar de que estos términos pertenezcan al vocabulario natural en uso o tenga una representación matemática clara (pertenencia, intersección, etc.).

Ambos principios tienen como fin que se utilice lo mas posible el lenguaje de la aplicación. Como resultado se tiene que el LEL posee un aspecto de hypertexto. Cada nodo está representado por el signo y su descripción. Los links lo determinan los signos que tienen una entrada en el LEL y se utilizan para definir otros. La Ilustración 6 muestra lo dicho:

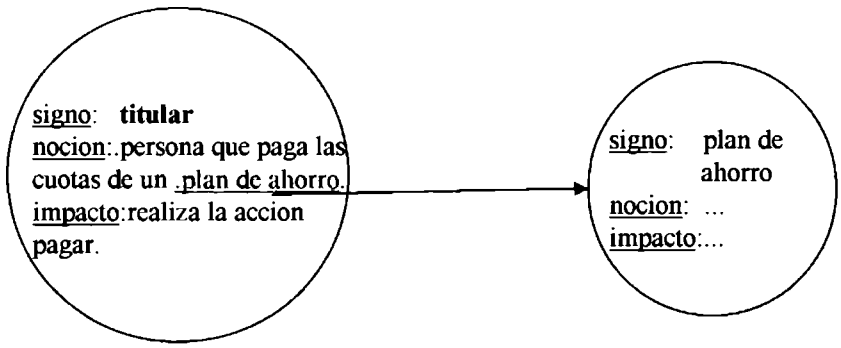


Ilustración 6

Por último, se ve la gramática que resume todo lo visto en esta sección:

(1) LEL $_{UofD}$ = {Entrada}$^{+}$
(2) Entrada = Titulo {Nocion}$^{+}$ {Impacto}$^{+}$

(3) Titulo = Signo / Titulo

(4) Titulo = Signo

(5) Nocion = $\{\text{Signo}, \neg\text{Signo}\}^{+}$

(6) Impacto = $\{\text{Signo}, \neg\text{Signo}\}^{\pm}$

(7) Signo $\cap$ $\neg$Signo = $\varnothing$

(8) $\neg$Signo $\in$ *NL*. *NL* es el subconjunto mas pequeño de palabras del lenguaje natural.

1.2.3.Construcción del LEL

A continuación se brindan los pasos a seguir para construir el LEL. El primero es:

1.2.3.1.Identificación de fuentes de información

Hay que determinar si se leerán documentos o se mantendrán entrevistas con los usuarios.

Cada uno tiene sus ventajas y desventajas. La ventaja de leer documentos es que es posible analizar en detalle el texto. Se puede leer y releer cuantas veces se quiera antes de identificar los símbolos. Sin embargo las charlas con los usuarios pueden ser más fructíferas. En primer lugar el caudal de información que se puede extraer de una entrevista es inmenso[1], en cambio, con los documentos el analista se encuentra enmarcado por la información escrita, no es imposible ampliar el texto. Otro punto a favor de las charlas es que cuando una persona se manifiesta oralmente, en forma inconsciente utiliza el principio de circularidad. En cambio cuando escribe, incorpora muchas más palabras, las que se convertien en sinónimos, y deben ser identificados.

Cualquiera sea la técnica de aproximación, hay que saber elegir las personas o documentos. Como guía para la elección de las personas se dan estos cuatro puntos:

- las más mencionadas dentro del entorno del sistema.
- las que toman las decisiones.
- los responsables del proyecto.
- los supervisores del proceso.

Los documentos que más información brindan son:

- descripciones textuales del sistema.
- manuales que definen los procedimientos operacionales.

1.2.3.2.Identificación de signos

Este proceso consiste en tomar todas las palabras o frases que parezcan estar fuera de contexto, o aquellas que más veces aparecen. La elección de los signos que parezcan estar fuera de contexto es porque el significado con el que se los utilizan es distinto al tradicional, así que es necesario definirlos dentro del ambiente de la aplicación. Automatizar este proceso es bastante costoso, por lo cual esta fase debe ser realizada en forma manual.

---

[1] El entrevistador debe adoptar una actitud pasiva y solo incentivar para que el entrevistado siga hablando.

La otra pauta para recolectar símbolos, es la cantidad de ocurrencias. Cuantas más veces aparece una palabra en el universo del discurso, indica que más significativa es, por lo que se necesita describirla de manera precisa. Esta parte si puede automatizarse de manera eficiente. Lo que se debe hacer es recorrer todo el texto contando las ocurrencia de cada palabra. Se desechan de la búsqueda artículos, preposiciones y pronombres. Y se toman las que más veces se repiten. Este algoritmo funciona bien en cuanto a palabras se refiere, para que también pueda reconocer frases hay que agregar cierta inteligencia adicional.

Se quiere poner de manifiesto que la idea detrás de la elección de símbolos es entender el lenguaje, de ningún modo se pretende elegir símbolos para determinar las funciones principales del sistema. Contrariamente al análisis de requerimientos tradicional en donde el analista identifica las funciones a partir de las entrevistas, el objetivo de estas entrevistas en familiarizarse con el lenguaje del dominio de la aplicación.

En cuanto a la recolección de signos, un elemento a tener en cuenta es que en la lista de signos obtenidos, pueden existir sinónimos. Estos sinónimos se pueden ir marcando en la medida de lo posible en esta fase. Sin embargo los sinónimos quedan determinados de manera precisa en la parte de definición, donde se detectan que las descripciones son similares para signos distintos.

1.2.3.3.Descripción de signos

Con la información obtenida y con la ayuda de un experto en el dominio, se procede a la descripción de cada uno de los signos. La misma consiste en redactar nociones e impactos, e identificar los sinónimos. A continuación se detallan una serie de reglas que pueden ayudar en esta proceso:

- Un signo puede tener una o más nociones y cero o más impactos.
- Cada noción e impacto debe ser descripto con oraciones breves y simples.
- Nociones e impacto para un signo pueden representar diferentes puntos de vista o pueden ser complementarios.
- Las oraciones breves y simples de las nociones e impactos deben responder a los principios de circularidad y de vocabulario mínimo.
- Para los signos que son sujeto de una oración, los impactos deben indicar las acciones que realiza.
- Para los signos que cumplen el rol de verbo, las nociones deben decir quien ejecuta la acción, cuando sucede y el proceso involucrado en la acción. Para los impactos se deben identificar las restricciones sobre la realización de la acción, que origina esta acción y que es lo que causa esta acción.
- Para los signos que son objetos de una oración, la noción debe identificar otros objetos con los cuales se relaciona. Y los impactos serán las acciones que se pueden realizar con este signo.

1.2.3.4.Validación

Este proceso consiste en verificar por parte del usuario que todo lo escrito sea correcto. Debe leer uno por uno los signos con sus respectivas nociones e impactos y verificar su correctitud.

Por parte del analista queda revisar que todos los signos se hayan definido. Luego de describir todas las entradas de la lista inicial, es posible que dentro de las nociones o impactos aparezcan nuevos términos que son necesarios describir. Esta validación consiste en leer todas las descripciones y verificar que todos los términos desconocidos ya se hayan definido.

Con esta última etapa se da por finalizada la construcción del LEL.

1.2.4.Ejemplo: Organizador de reunión

En esta sección se muestra un ejemplo concreto de LEL. Se ve una descripción del problema y el LEL obtenido.

En el párrafo siguiente se muestra una breve descripción que sirvió para elaborar la lista de signos. Luego, con ayuda de los usuarios se obtuvo más conocimiento y se definieron los términos.

> *Debe ser construido un organizador de reuniones. El mismo debe determinar para un pedido de reunión, una fecha y lugar, de forma tal que la mayoría de los invitados propuestos puedan acudir.*
>
> *El proceso de organización de una reunión es iniciado por una persona conocida como iniciador de reunión. Él propone un rango de fechas entre las cuales la reunión debe llevarse a cabo y elige a las personas que participarían de la reunión (participantes propuestos). La reunión puede ser profesional o privada. Una reunión profesional sólo puede tener lugar durante horas de oficina. Y una reunión privada sólo puede llevarse a cabo fuera del horario de oficina.*
>
> *Cada participante tiene una agenda personal. Basado en ellas, el organizador de reuniones deriva un conjunto de fechas durante las cuales los participantes no pueden acudir (conjunto de exclusión) y el conjunto de fechas que los participantes prefieren (conjunto de preferencia). Usando el conjunto de exclusión y el conjunto de preferencia el organizador de reuniones planea la reunión. Él debe hallar un lugar y una fecha donde todos los participantes puedan asistir y, de ser posible, dentro de todos los conjuntos de preferencia. Tres cosas pueden suceder. Primero, la fecha se puede encontrar, entonces, todos los participantes deben ser informados. Segundo, puede ocurrir un conflicto débil de fechas. O tercero, puede ocurrir un conflicto fuerte de fechas. Un conflicto débil de fechas ocurre cuando alguna fecha puede ser hallada dentro del rango de fechas y fuera de los conjuntos de exclusión, pero no se puede hallar dentro de la intersección de todos los conjuntos de preferencia. Y un conflicto fuerte de fechas ocurre cuando ninguna fecha se puede hallar dentro del rango de fechas y fuera de todos los conjuntos de exclusión. En los últimos dos casos un replaneamiento de la reunión se debe hacer aplicando alguna política de resolución de conflictos de fecha.*

Las distintas entradas del LEL que se obtuvieron son las siguientes. Cabe destacar que los signos en rojo son aquellos que están definidos.

Signo	**Organizador de reuniones**
Noción	1. Sistema que soporta la organización de reuniones

	2. Maneja varios pedidos de reuniones
Impacto	1. Determina, para cada pedido de reunión, una fecha de reunión y un lugar de reunión, de forma tal que la mayoría de los participantes propuestos puedan participar. 2. Reduce el trabajo de organizar reuniones donde los participantes potenciales están distribuidos en diferentes lugares. 3. No debe quebrantar restricciones físicas.

Signo	**Pedido de reunión**
Noción	1. Son hechos por usuarios autorizados para concertar una reunión.
Impacto	1. Pueden ocurrir muchos en paralelo. 2. Se pueden superponer en tiempo o espacio.

Signo	**Reuniones**
Noción	1. Pueden ser reuniones profesionales o reuniones privadas. 2. Cada una tiene una fecha y un lugar.
Impacto	1. La mayoría de los participantes propuestos efectivamente participan de ella. 2. El proceso de concertarla es originado por un iniciador de la reunión.

Signo	**Fecha de reunión/ Fecha de reunión propuesta**
Noción	1. Es la fecha mas conveniente para todos los participantes para acudir a la reunión. 2. Se define por un par (fecha del calendario, periodo de tiempo) 3. Esta dentro del rango de fechas y fuera de todos los conjuntos de exclusión. 4. Idealmente pertenece a tantos conjuntos de preferencia como sea posible.
Impacto	1. Puede existir alguna dependencia con algún lugar de reunión 2. Cuando ninguna fecha se puede encontrar, un conflicto de fechas ocurre.

Signo	**Lugar de reunión/ Sala de reunión**
Noción	1. Es el lugar mas conveniente para todos los participantes que acuden a la reunión.
Impacto	1. Debe estar disponible para la fecha de reunión. 2. Debe idealmente pertenecer a alguno de los lugares preferidos por los participantes importantes en la manera que sea posible.

	3. Puede existir alguna dependencia con la fecha de reunión. 4. Una nueva ronda de negociaciones puede ser requerida cuando ninguna sala se puede encontrar.

Signo	**Iniciador de reunión**
Noción	1. Puede ser una de los participantes o algún representante
Impacto	1. Define el rango de fechas de la reunión 2. Pide a todos los potenciales participantes de la reunión sus conjuntos de exclusión y preferencia. 3. Pide a los participantes activos proponer cualquier equipamiento especial requerido en el lugar de reunión. 4. Pide a los participantes importantes sus preferencias sobre los lugares de reunión.

Signo	**Participantes propuestos / potenciales participantes / participantes potenciales de la reunión / participantes interesados**
Noción	1. Persona que se supone irá a la reunión. 2. Cada uno tiene un conjunto de información asociada que puede variar (agenda personal).
Impacto	1. La información de la reunión debería difundirse tan pronto como sea posible hacia todos ellos.

Signo	**Agenda personal**
Noción	1. Un conjunto de información de los participantes propuestos. 2. Esta información puede ser expresada de distintas formas.
Impacto	1. Basada en ella, el conjunto de exclusión y el conjunto de preferencia se derivan.

Signo	**Conjunto de exclusión**
Noción	1. Un conjunto de fechas en las cuales los participantes potenciales no pueden acudir a la reunión. 2. Esta contenido en algún intervalo de tiempo prescrito por el iniciador de la reunión (rango de fechas).
Impacto	1. Esta basado en la agenda personal. 2. Evaluando el de cada participante propuesto, junto con el conjunto de preferencia, el organizador de reunión genera una fecha de reunión.

Signo	**Rango de fechas**

Noción	1. El intervalo de tiempo prescrito por el iniciador de reunión donde los conjuntos de exclusión y preferencia deben estar contenidos.
Impacto	1. La fecha propuesta de reunión y fecha de reunión deben estar dentro de este rango de fechas.

Signo	**Conjunto de preferencia**
Noción	1. Un conjunto de fechas en las cuales un participante potencial puede acudir a la reunión. 2. Esta contenido en algún intervalo de tiempo prescrito por el iniciador de reunión (rango de fechas). 3. Prioridades explícitas entre fechas se puede introducir.
Impacto	1. Se basa en la agenda personal. 2. Evaluando el de cada participante propuesto, junto con su conjunto de exclusión, el organizador de reuniones genera una fecha de reunión.

Signo	**Participante**
Noción	1. Una persona que ira a una reunión
Impacto	1. Se les permite modificar su conjunto de exclusión, conjuntos de preferencia y/o lugar preferido antes que la fecha de reunión/lugar de reunión sea propuesto.

Signo	**Equipamiento especial**
Noción	1. Equipo que un participante activo necesita en el lugar de reunión
Impacto	1. El iniciador de reunión pide a los participantes activos que provean sus requisitos.

Signo	**Conflicto de fechas**
Noción	1. Ocurre cuando una fecha propuesta de reunión no se puede hallar. 2. Puede ser un conflicto fuerte de fecha o un conflicto débil de fecha.
Impacto	1. Se debe resolver usando una política de resolución de conflictos.

Signo	**Participantes importantes**
Noción	1. Son los participantes de la reunión.
Impacto	1. Pueden plantear su preferencias sobre el lugar de reunión cuando se lo pide el iniciador de reunión

Signo	**Conflicto fuerte de fecha**
Noción	1. Ocurre cuando ninguna fecha se puede hallar dentro del rango de fechas y fuera de todos los conjuntos de exclusión.
Impacto	1. Genera un replaneamiento de la reunión, con la aplicación de algunas políticas de resolución de conflictos de fechas.

Signo	**Conflicto débil de fecha**
Noción	1. Ocurre cuando se puede encontrar alguna fecha dentro del rango de fechas y fuera de todos los conjuntos de exclusión, pero ninguna dentro de todos los conjuntos de preferencia.
Impacto	1. Genera un replaneamiento de la reunión, con la aplicación de algunas políticas de resolución de conflictos de fechas.

Signo	**Representante de participante**
Noción	1. El que es pedido por un participante, a través de delegación, representarlo en la reunión.
Impacto	

Signo	**Participante activo**
Noción	1. Es un participante.
Impacto	1. Provee requerimiento de equipamiento especial cuando se lo pide el iniciador de reunión.

Signo	**Planear una reunión / replanear una reunión**
Noción	1. Actividad que debe ser realizada por el organizador de reuniones, bajo las restricciones de participantes.
Impacto	1. Una fecha y un lugar para la reunión son definidos. 2. Todos los participantes deben ser informados de la fecha y del lugar definidos para la reunión.

Signo	**Restricciones externas**
Noción	1. Son restricciones que no están bajo la influencia del organizador de reuniones. 2. Son tomadas en cuenta luego de que una fecha de reunión y un lugar de reunión han sido propuestos.
Impacto	1. Como consecuencia de ellas, un replaneamiento de la reunión puede ser

	necesario; a veces la reunión es necesaria; a veces la reunión puede ser cancelada.

Signo	**Reunión más importante**
Noción	1. Una reunión que es más importante que otra alocada en un lugar y fecha y está necesita su fecha y lugar la puede tomar
Impacto	1. La fecha y lugar de la reunión original debe ser cambiada o la reunión puede ser cancelada

Signo	**Restricciones de los participantes**
Noción	1. Es la información expresada en los conjuntos de exclusión, conjunto de preferencia y/o lugar de preferencia. 2. Se pueden expresar de distintas formas. 3. Algo (o toda) de esta información puede ser modificada por los participantes antes que la fecha de reunión o lugar de reunión sea propuesta.
Impacto	1. Cuando algo de esta información es modificada por algún participante antes que la fecha de reunión o lugar de reunión sea propuesta, un replaneamiento de la reunión se necesita.

Signo	**Políticas de resolución de conflictos**
Noción	1. Políticas usadas para resolver conflictos de fechas. 2. Son manifestadas por los cliente. 3. Son conocidas por el organizador de reunión para solucionar los conflictos de fecha.
Impacto	1. La política adoptada debe ser una de las siguientes: a) El iniciador de reunión extiende el rango de fechas; b) Algunos participantes quitan algunas fechas de sus conjuntos de exclusiones; c) Algunos participantes desisten de concurrir a otras reuniones; d) Algunos participantes agregan nuevas fechas a sus conjuntos de preferencia.

Signo	**Cliente / usuario autorizado**
Noción	1. El que está autorizado a hacer requerimientos de reuniones, independientemente de su paradero. 2. Tiene una agenda personal
Impacto	1. Define politicas de resolución de conflictos.

Signo	**Reglas privadas**
Noción	1. Reglas que no permiten a un participante ordinario conocer restricciones impuestas por otros participantes.
Impacto	1. No pueden ser quebrantadas por el organizador de reuniones.

Signo	**Restricciones físicas**
Noción	1. Restricciones que son impuestas por reglas y leyes concernientes al mundo real. 2. Son, por ejemplo: a) una persona solo puede concurrir a una reunión por vez. b) lugares de reunión no pueden ser usados por mas de una reunión por vez.
Impacto	1. No pueden ser quebrantados por el organizador de reunión cuando organiza reuniones.

Signo	**Reuniones profesionales**
Noción	1. Reuniones que solo se pueden realizar durante horario de oficina.
Impacto	

Signo	**Reuniones privadas**
Noción	1. Reuniones que solo se pueden realizar en horario de descanso
Impacto	

Signo	**Lugar preferido**
Noción	1. Es el lugar que algún participante importante prefiere para que la reunión tome lugar en él.
Impacto	

Signo	**Participación parcial**
Noción	1. Situación en la cual los participantes sólo pueden concurrir a parte de la reunión.
Impacto	1. Esto sucede como consecuencia de un conflicto de fechas que no es posible resolver completamente.

1.2.5.Herramienta que soporte LEL

La única herramienta de la que se tiene información, que se utilizó para soportar el modelo del LEL, es un procesador de textos común y corriente. En [Leite 90] se analiza el lenguaje de la biblioteca de la Pontificia Universidade Católica do Rio do Janeiro y toda la información recogida era volcada en un editor de textos. Algo parecido a como se hace en el ejemplo de la sección 1.2.4, para cada término se identifican las nociones e impactos y aquellos términos dentro de nociones e impactos que estén también definidos, deben marcarse de alguna forma (negrita, subrayada o algún color distinto de negro). Las facilidades que provee el editor para la creación del LEL, son las mismas facilidades que provee cualquier editor de textos contra la edición de texto en una máquina de escribir o en forma manual. Permite insertar líneas de descripciones entremedio de otras ya existentes, permite ordenar todas las entradas alfabéticamente por el signo. Pero en cuestión de navegación no ofrece ninguna ventaja. Si mientras se lee una descripción aparece un término el cual también esta definido, hay que buscar entre todas las entradas al signo, para ver su descripción. Tampoco ayuda de ninguna manera el proceso de validación. El analista debe releer el texto asegurando su consistencia. A continuación se muestra un ejemplo de como podría utilizarse un procesador de textos para volcar la información el LEL.

Habría que crear un tabla con dos columnas. En la primera se coloca el término y en la otra las nociones e impactos. Y como se menciona en el párrafo anterior, si algún símbolo de las descripciones es algún término definido, debe marcárselo de alguna manera. Por último hay que ordenar el cuadro por la primera columna, para poder ubicar las entradas del LEL rápido. Lo que se acaba de describir se ve de la siguiente manera.

Organizador de reuniones	Nocion 1. Sistema que soporta la organización de reuniones 2. Maneja varios pedidos de reuniones Impacto 1. Determina, para cada pedido de reunión, una fecha de reunión y un lugar de reunión, de forma tal que la mayoría de los participantes propuestos puedan participar. 2. Reduce el trabajo de organizar reuniones donde los participantes potenciales están distribuidos en diferentes lugares. 3. No debe quebrantar restricciones físicas.
Reuniones	Nocion

	1. Pueden ser reuniones profesionales o reuniones privadas. 2. Cada una tiene una fecha y un lugar. Impacto 1. La mayoría de los participantes propuestos efectivamente participan de ella. 2. El proceso de concertarla es originado por un iniciador de la reunión.

Ilustración 7

1.3.Escenarios

Los escenarios son bien reconocidos como una importante estrategia para especificar el comportamiento del soft, como así también para delimitar la frontera entre el entorno y el programa.

1.3.1.Introducción

Un escenario es una descripción parcial y concreta del comportamiento de un sistema en una determinada situación. La utilización de escenarios implica identificar distintas situaciones y describir la acción a llevar a cabo. Los mismos son de gran ayuda en el momento de especificar requerimientos; y su rol principal es el de permitir la comunicación entre expertos de software y del dominio, y analizar aspectos específicos de un sistema describiéndolo en forma concreta. La ventaja de los escenarios sobre cualquier otro método de especificación de requerimientos es que los escenarios guardan una gran similitud a la forma en que los seres humanos entienden y describen los problemas.

Los escenarios describen actores, objetivos y episodios. Los actores son las entidades que hacen uso del sistema para satisfacer cierta necesidad (objetivo). Para ello, realizan acciones (episodios) las cuales involucran la actividad de alguna función del sistema.

Es una descripción parcial, porque no necesita describir todas las características de las entidades involucradas, sólo se describen aquellos elementos que son relevantes para el escenario. Por ejemplo, en un escenario que describe un auto atravesando una intersección, no tiene sentido mencionar los ocupantes del auto o los negocios junto a las calles. Lo importante es como se comportará en relación a los otros autos que también intentan atravesar la avenida.

Y es una descripción concreta de una determinada situación, ya que siguiendo con el ejemplo del cruce, se analiza el caso específico de un vehículo que se acerca a un cruce e intenta atravesarlo. En el escenario se analizan las distintas variantes en función a los otros autos que se acercarán al cruce.

Las descripción de los escenarios está formada de dos elementos. El comportamiento (behavior) y la situación (situation). Los dos son una sucesión de estados que se expresan utilizando los mismos constructores. Ellos son: secuencialidad (un estado seguido de otro), concurrencia total o parcial (dos eventos pueden realizarse simultáneamente), repetición, permutación, o inclusivo, o exclusivo (evento 1 o evento 2, pero no ambos), negación (evento 3 no ocurrirá), etc. Por ejemplo el comportamiento del auto atravesando la calle puede especificarse como una secuencia de estados: auto frente a la intersección, auto en medio de la intersección, auto fuera de la intersección. Mas allá de que situación y comportamiento se describan de la misma forma, hay una diferencia semántica. La situación describe el estado inicial a partir del cual se realizarán las acciones. El comportamiento representa todas las acciones que se llevan a cabo para la situación presentada.

En lo que concierne a los elementos generales de los escenarios, hay dos más que mencionar para concluir con esta introducción.

En primer lugar es el formato en el cual se genera. El mismo es muy variado. Pueden ser hechos en lenguaje natural, como los uses cases en [Jacobson 94] y [Jacobson 94b], los cuales se ven en la sección 1.3.2. Pueden ser storyboards, una descripción gráfica en donde las acciones se identifican con distintos cuadros como si fuera una historieta. O bien pueden ser diagramas de interacción entre objetos.

El segundo elemento es la organización de los escenarios. Si bien cada escenario está autocontenido, para tener una visión global de todo el sistema o para clasificarlos, (ya que la cantidad puede ser muy grande) los escenarios deben ser agrupados, organizados o compuestos.

Hay cinco conceptos de organización. El primero consiste en agrupar los escenarios que tratan de una tarea particular o distintas alternativas para una misma tarea. Por ejemplo, se puede hablar de atravesar una intersección sin semáforos, con semáforos, con cruce a nivel, yendo por una avenida, cruzando la avenida. Todos se refieren a cruces, pero en distintas situaciones.

El segundo se refiere a la evolución de escenarios. Los escenarios evolucionan y cambian junto con la evolución del proyecto. La idea de este trabajo es además de seguir toda la evolución de los escenarios, seguir la evolución de los escenarios y del LEL en forma conjunta. Para un cierto momento del desarrollo del soft, poder ver como está compuesto el LEL y los escenarios.

El tercero es la elaboración de escenarios. Un escenario muy complejo puede ser elaborado o descompuesto en otros más simples. Se podría tener un solo escenario que indique como atravesar un cruce teniendo en cuenta semáforos, cruces de ferrocarriles, avenidas y ante la complejidad crear todos escenarios distintos para cada caso particular.

El cuarto se basa en componer escenarios. Aprovechando la semántica del "o" lógico se pueden relacionar escenarios formando cursos alternativos. Se parte del hecho de atravesar un cruce, pueden existir o no semáforos, si no hay semáforos, se puede tratar de cruzar una avenida o por la avenida. Si hay semáforos, puede que se esté esperando detenido que de luz verde o la luz verde está mientras se aproxima.

Y por último, la composición propiamente dicha de escenarios, en donde inversamente a lo planteado en el punto tercero, distintos escenarios se agrupan para formar uno más complejo. Se puede hablar de llegar de un punto a otro de una ciudad y a lo largo del camino se encuentran cruces de calles de distintas características.

Los siguientes gráficos, ilustran los conceptos expuestos.

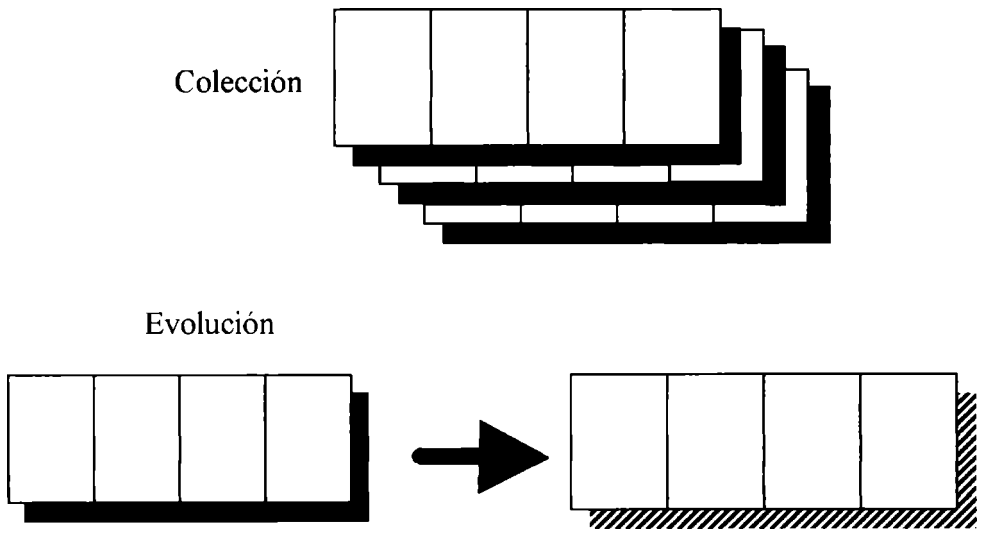
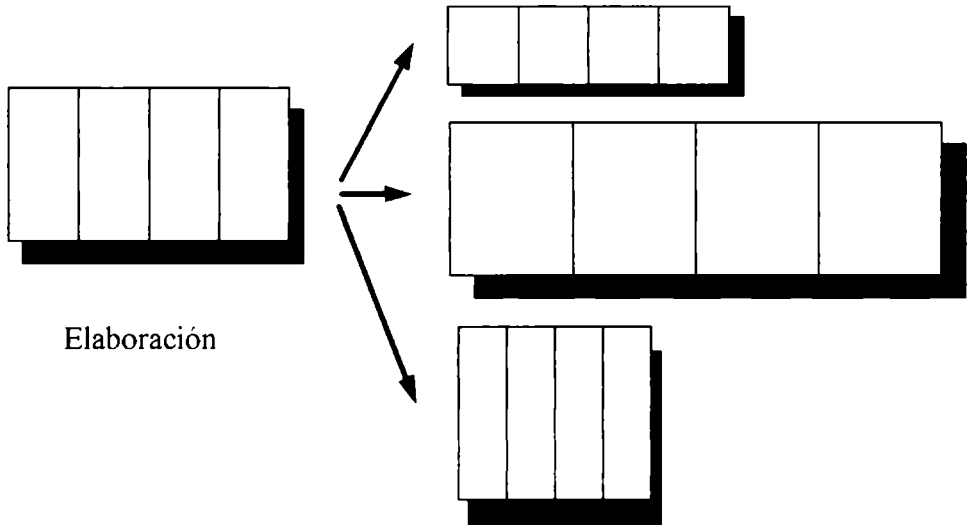
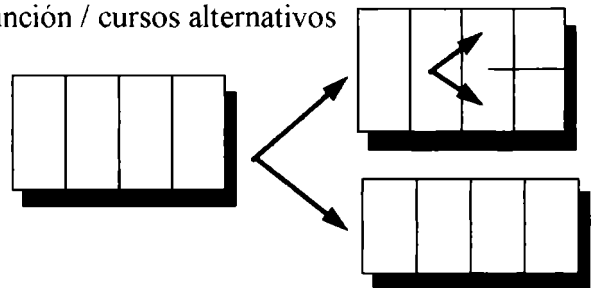


Ilustración 8



Elaboración

Disyunción / cursos alternativos



Composición

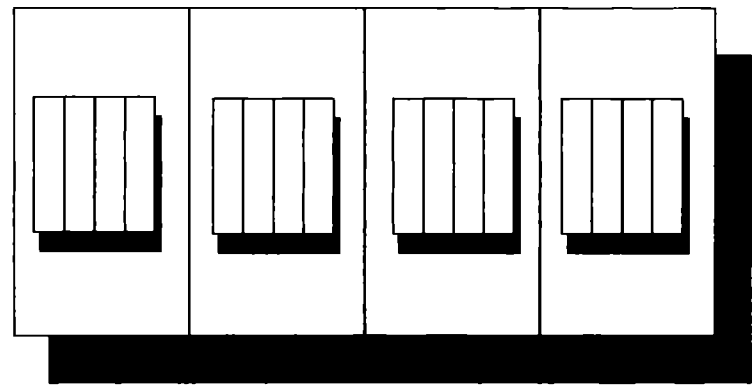


Ilustración 9

1.3.2.Modelos existentes

A continuación se analizan modelos de escenarios propuestos por diferentes autores.

1.3.2.1.Uses Cases de Jacobson

Dentro de su metodología orientada a objetos, Jacobson propone el modelo de uses cases ([Jacobson 94] y [Jacobson 94b]) para capturar la funcionalidad del sistema.

Un use case es la descripción de un tipo de uso del sistema. El conjunto de todos los uses cases representa la funcionalidad completa del soft. Si bien, como se ve a continuación hay muchas similitudes entre uses cases y escenarios, Jacobson fija las diferencias que existen entre estos dos elementos. Él dice que los escenarios normalmente representan instancias de uses cases y no tienen ningún equivalente a las uses cases clases. Como se ve en la sección 1.3.1 "los escenarios son descripciones parciales y **concretas** ..." y **un** use case representa el conjunto de todos los usos de una misma funcionalidad del sistema, así que un escenario sería un uso en particular del sistema, un elemento del conjunto de un use case.

Para ver las similitudes entre escenarios y use cases, se muestra lo que dice Jacobson de los use cases. Él plantea que el modelo de use cases define el comportamiento del sistema (una de las dos componentes fundamentales de los escenarios). Se describe el entorno del sistema (situación como se llama en los escenarios) a través de los distintos usuarios que operan el sistema a través de los use cases.

1.3.2.1.1.Modelo de use cases

El modelo de use cases es un grafo con dos tipos de nodos, nodos actores y nodos use cases, y con un nombre, el nombre del sistema. Cada nodo actor tiene un nombre y una clase. Los nombre de los nodos actores son únicos. Por su parte, cada nodo use case tiene también un nombre y una clase. Los nombres de los nodos use cases, también son únicos.

Un nodo actor tiene al menos un arco hacia un nodo use case, y un nodo use case tiene al menos un arco hacia un nodo actor. Estos arcos se denominan arcos de comunicación.

Una instancia de un actor puede crear instancias de use cases, y una instancia de use cases obedece a su clase. Un arco de comunicación entre un nodo actor y un nodo use case significa que un estímulo ha sido enviado entre una instancia de la clase actor y una de la clase use case o entre instancias de la clase use case.

Los actores son objetos que residen fuera del modelo del sistema. Representan todo lo que necesita intercambiar información con el sistema. Nada más fuera del sistema tiene algún impacto en él. Los actores pueden ser humanos u otro sistema.

Se hace una distinción entre actores y usuarios. Un usuario es un humano que usa el sistema, en cambio, un actor representa un rol específico que un usuario puede jugar. Los actores son instancias de una clase, y los usuarios son algún tipo de recursos que implementan estas instancias. El mismo usuario puede así actuar como instancias de diferentes actores.

A continuación se considera el ejemplo de un cajero automático que realiza operaciones sencillas tales como extraer, depositar, dar el balance de una cuenta. Este cajero está conectado con la central del banco quien es en realidad quien verifica que las acciones del usuario sean correctas (por ejemplo que su número de identificación sea correcto, que pueda extraer dinero, etc.).

Para este ejemplo se pueden identificar tres tipos de actores. Se tiene el cliente del banco, quien realiza las transacciones. También está el operador quien hace el mantenimiento del sistema. Y por último, la computadora del banco que interactua con el cajero para realizar las operaciones.

A continuación se muestra el use case que modela la extracción de dinero.

Curso normal

Se muestra en la pantalla un mensaje que indica que el cajero esta disponible:

El cliente inserta su tarjeta en el cajero

El cajero lee el código de la tarjeta y chequea si es aceptable

Si la tarjeta es aceptable el cajero pide el número personal de identificación (pin)

El cajero espera el pin:

El cliente entra el pin

Si el pin es correcto el cajero pide el tipo de transacción a realizar.

El cajero espera por el tipo de transacción:

El cliente selecciona extracción de dinero

Si el cliente puede extraer, el cajero muestra la información de la cuenta del cliente y pide cantidad de dinero a extraer.

El cajero espera por la cantidad de dinero a extraer

El cliente selecciona la cantidad de dinero a extraer

Si esa cantidad es posible, se entrega el dinero, se imprime la transacción y se devuelve la tarjeta.

Cursos alternativos

La tarjeta no es aceptada:

Se devuelve la tarjeta

El pin es incorrecto

Se muestra un error y se da una nueva oportunidad. Si falla se retiene la tarjeta.

La extracción no es permitida

Se muestra un mensaje y se devuelve la tarjeta

La cantidad de dinero a extraer no es permitida

Se muestra un mensaje y se permite ingresar una nueva cantidad

Cancelar la operación

El cliente puede cancelar la operación en cualquier momento en que el cajero le esta pidiendo el ingreso de información. Se da por finalizada la transacción y se devuelve la tarjeta

A partir de esta descripción se puede analizar la funcionalidad del sistema, sirviendo como una herramienta de comunicación entre ingenieros y usuarios.

Ahora bien, los sistemas reales puede contener un gran número de use cases. Un sistema mediano puede incluir entre 10 y 60 use cases, así que se necesita de una manera de relacionar use cases para evitar redundancia. Esto se logra por medio de dos tipos de arcos: usa y extiende.

1.3.2.1.2.Relación usa

Es normal encontrar use cases con descripciones similares, para evitar redundancia y fomentar el reuso, se necesita una manera de extraer la descripción similar y compartirla entre los diferentes casos de uso. Se dice que estas descripciones son descripciones de use cases abstractos y que los use cases originales (los que comparten la descripción) son use cases concretos. Se llaman use cases abstractos a los primeros porque no serán instanciados por sí solos, sólo se harán para describir partes compartidas entre otros use cases. En cambio los use cases concretos serán los que sí serán instanciados. En el ejemplo del cajero automático, muchos use cases compartirán la secuencia que chequea la tarjeta y el pin, por lo que se puede crear un use case abstracto. El siguiente diagrama muestra la relación usa.

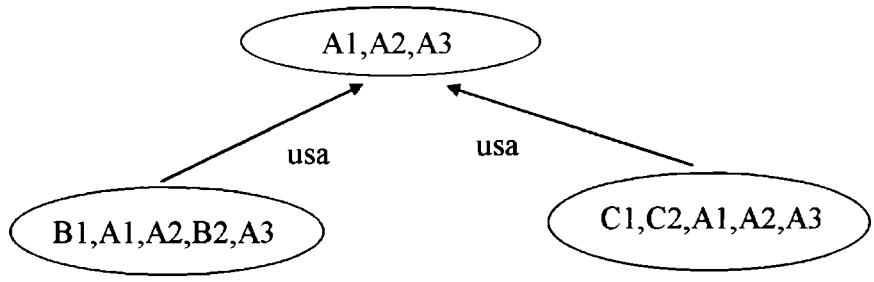


Ilustración 10

Como se puede ver en el diagrama la parte común (A1, A2 y A3) se puede intercalar a gusto por los use cases concretos. Lo único que se debe respetar es el orden, A1 primero, entre A1 y A3, A2 y por último A3.

1.3.2.1.3.Relación extiende

Un arco extiende desde un use case A hacia un use case B significa que se interpretará el use case B, en algún momento se interrumpirá, se interpretará A y luego terminará con B.

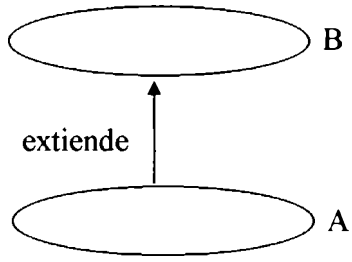


Ilustración 11

La utilización de la relación extiende se deja para indicar partes opcionales, cursos complejos y alternativos, subsecuencias que sólo se ejecutan en ciertos casos e intersecciones de diferentes use cases en otros.

Para fijar las diferencias entre la relación usa y extiende se ve el siguiente ejemplo.

1.3.2.1.4.Diferencia entre relación usa y extiende

Para fijar la diferencia entre la relación usa y extiende se supondrá un modelo de use cases con tres nodos. El primero es transacción: el retiro o deposito de dinero. Luego se tiene estadísticas de transacciones: cuanto dinero se depositó o extrajo. Y por último supervisor de transacciones, por medio del cual, los desarrolladores espían como es usado el cajero. El modelo quedaría de la siguiente forma:

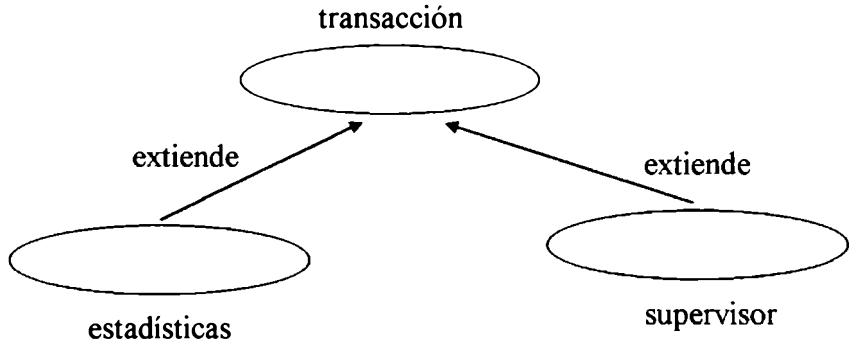


Ilustración 12

La causa de por que se utilizó la relación extiende es que la ejecución de estadísticas y supervisor nace como una extensión a la transacción original. Se podría

modelar con la relación use como se ve a continuación, sin embargo esto va en contra de la idea de extensibilidad.

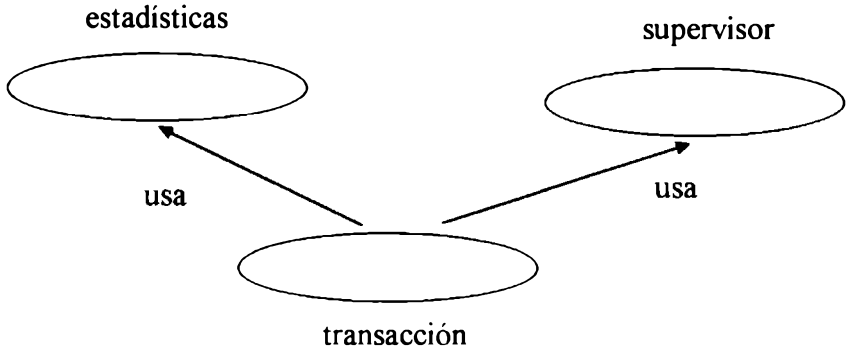


Ilustración 13

Y otra posibilidad sería agregar un nuevo nodo que concentra la funcionalidad de estadísticas y supervisor, pero tiene el costo de agregar un nodo más al grafo.

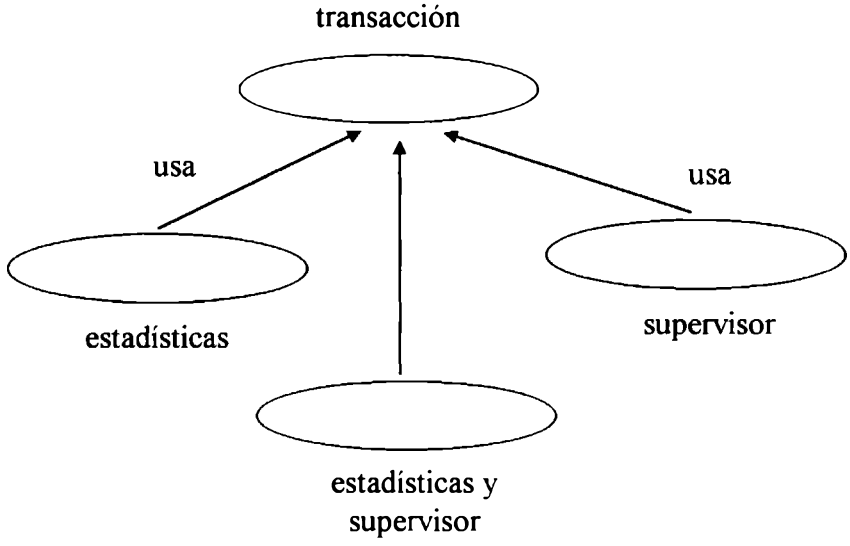


Ilustración 14

1.3.2.2.OBA

Se propone la descripción de escenarios a través de scripts. Un script es una descripción estructurada de un uso típico del sistema. Se forman realizando un contrato entre dos roles. El primer rol, iniciador, colabora con el segundo, participante, para realizar un paso de la tarea completa. El iniciador realiza una acción, responsabilidad y el participante responde con otra acción, el servicio correspondiente. El siguiente cuadro muestra las relaciones posibles entre los objetos.

Iniciador	Acción	Participante	Servicio
Iniciador 1	notifica	Participante 1	Participante 1 puede ser notificado
Iniciador 2	provee información a	Participante 2	Participante 2 acepta la información
Iniciador 3	pide información desde	Participante 3	Participante 3 puede proveer información
Iniciador 4	pide servicio de	Participante 4	Participante 4 puede proveer servicio

Ilustración 15

Cada script contiene: nombre, autor, versión, precondición (estado del sistema para que ese script suceda), postcondición (estado final del sistema al finalizar el script), trace (área de actividad del dominio a la que pertenece ese script).

1.3.2.2.1.Uso de los escenarios en la metodología

Estos scripts surgen en la etapa de análisis de requerimientos para capturar la funcionalidad del sistema. En la etapa de diseño se utilizan los scripts para encontrar los objetos del sistema, sus responsabilidades y colaboraciones con el resto de los objetos. Finalmente se usan las pre y post condiciones para determinar el ciclo de vida completo de un objeto.

1.3.2.3.ICM [Potts 94]

El ICM es un modelo de hipertexto dinámico que captura el proceso de elaboración de los requerimientos. En este modelo se denomina **stackholder** al usuario y a cualquiera que puede intercambiar información sobre el sistema, sus restricciones de implementación o del dominio del problema.

El modelo presenta tres fase:

1.- documentación de requerimientos: los stackholder describen los requerimientos. Cada requerimiento es un nodo en el hipertexto.

2.- discusión de los requerimientos: el stackholder revisa los requerimientos a través de preguntas. Él puede encontrar ambigüedades, inconsistencias o requerimientos incompletos.

3.- evolución de los requerimientos: surgen cambios que refinan requerimientos o agregan nuevos.

En el ICM, los escenarios son considerados como descripciones de una o mas transacciones finales involucrando al sistema y su ambiente. Son representados usando dos niveles:

1.- Familias de escenarios: Una familia de escenarios permite tener una descripción general de un escenario particular y sus subescenarios (similar a la relación uses de Jacobson).

2.- Episodios o fases: Los episodios son secuencias de acciones mas detalladas.

1.3.2.3.1.Ejemplo

El trabajo analiza el problema del meeting scheduler. A partir de la documentación se encontraron los siguientes escenarios:

1.Sin conflictos
2.Un participante demora en contestar
3.Se agrega un participante más tarde
4.Retirar un participante
5.Los participantes cambian las preferencias de fechas antes que se organice la reunión
6.Conflicto de fechas: el iniciador modifica el rango de fechas posibles
7.Conflicto de fechas: los participantes modifican sus rangos de fechas
8.Conflicto de fechas: algunos participantes desisten de la reunión
9.Conflicto de fechas: los participantes excluyen días
10.Conflicto con el salón de reunión
11.Se interpone una reunión mas importante
12.Surgen conflictos posteriores al arreglo de la reunión
13.Se cancela

La descripción del escenario 2, es la siguiente:

1.El iniciador determina los participantes
2.El iniciador determina el rango de fechas
3.Se envían los aviso de reunión
4.Algunos participantes responden
5.El iniciador determina que se ha terminado el tiempo de espera
6.El scheduler da por finalizada la espera y elige el día sin tener en cuenta a los participantes que no respondieron

Analizando este escenario, surgen algunas dudas sobre la fecha de espera a las contestaciones. Una pregunta posible sería saber si existe una relación entre el día elegido y el rango de fecha, es decir cual es exactamente el último día en que los participantes pueden responder? Esto da lugar a modificaciones. En la Ilustración 16 se muestra la evolución de las modificaciones.

En [Potts 94], los autores explican que el 55% de los cambios realizados a los requerimientos del sistema fueron hechos a través del análisis de escenarios. Según su experiencia, muchas de las preguntas sobre los requerimientos de un sistema no pueden ser fácilmente analizadas si no es a través de escenarios.

1.3.2.4.ICM, una nueva propuesta [Potts 95]

En este trabajo se especifica en forma más detallada el concepto de escenarios, se identifican partes que los componen y se presentan estrategias para definirlos.

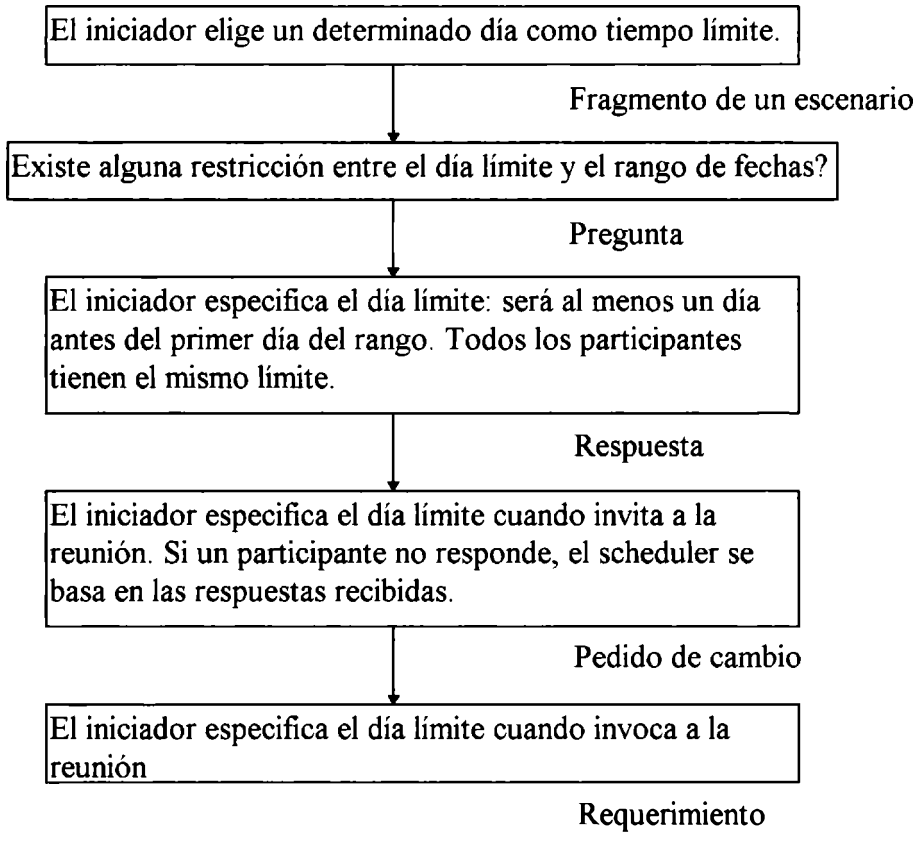


Ilustración 16

1.3.2.4.1.Definición y uso de escenarios

Un escenario es una descripción narrativa de un uso concreto del sistema. Describe la ejecución de una parte de la funcionalidad del mismo. Los escenarios tienen actores con objetivos (condiciones a ser alcanzadas). Estos objetivos pueden no alcanzarse por determinados obstáculos. Estos obstáculos pueden ser condiciones del sistema o porque se ha alcanzado algún otro objetivo (conflictivo). Los objetivos y obstáculos están representados por episodios. Un episodio es un conjunto de acciones asignadas a determinados actores. Un actor no necesariamente es una persona o agente fisico, un actor representa un rol dentro del sistema.

El esquema propuesto de escenarios es el siguiente:

Escenario: nombre del escenario
Settings:
Background: información, estado o situación del sistema que indica los objetivos de los actores.
Roles(actores): Cada uno de los actores intervinientes.
Narrativa:
Conjunto de episodios: Cada episodio esta descripto por: objetivos, obstáculos (opcional), acciones y logros.

El hecho de considerar obstáculos fuerza a los diseñadores a pensar en soluciones flexibles y robustas para situaciones del sistema no idealizadas (por ejemplo: errores cometidos por el usuario o usos del sistema en una forma no prevista durante el análisis y diseño).

1.3.2.4.2.Cómo encontrar escenarios?

Debido a que para un determinado sistema se pueden definir un número ilimitado de escenarios, a continuación se propone una estrategia para acotar el desarrollo de los mismos. Los escenarios deben ser derivados de los objetivos del sistema y de los potenciales obstáculos que bloquean estos objetivos.

La estrategia propone los siguientes pasos:

1. identificación de los objetivos: se deben tener en cuenta solo los objetivos funcionales y no los objetivos externos como pueden ser de mantenimiento o de optimización.
2. descomposición de los objetivos: se debe armar una jerarquía de objetivos. Es una tarea top-down en donde se deben descomponer los objetivos en subobjetivos que los cumplan. Para que un objetivo se cumpla sus subobjetivos se deben cumplir ya sea a través de la disyunción o conjunción.
3. operalización de los objetivos: consiste en establecer cuales son las acciones para alcanzar los objetivos y que actores las realizarán.
4. identificación de los obstáculos: para encontrar obstáculos se deben buscar fallas y errores posibles de los usuarios, que pasa si el usuario se olvida de mantener su agenda actualizada?, (recordando el ejemplo del meeting scheduler). Confusiones, que pasa si se confunden de personas y se invita a una persona no deseada? Falta de disponibilidad de recursos, la sala de reunión se pone fuera de servicio porque es necesario redecorarla. Las entrevistas y observaciones en el lugar del trabajo pueden proveer información sobre obstáculos y formas de reponerse a ellos.

A partir de este momento se derivan los escenarios. Para limitar el número de escenarios y que estos cubran la funcionalidad del sistema se debe tener en cuenta:

1. se deben cubrir todos los objetivos, de esta forma se cubre la funcionalidad completa del sistema.
2. respetar las dependencias en la jerarquía de objetivos: los episodios en los escenarios solo pueden ocurrir en cierto orden, por ejemplo no tiene sentido considerar casos en los cuales se está organizando las fechas de una reunión antes de que esté invocada.
3. considerar todos los tipos de obstáculos: el conjunto de escenarios debe contener una instancia de cada tipo de obstáculo encontrado. Cada obstáculo es considerado sólo en un escenario.
4. construir escenarios explorando interacciones de obstáculos interesantes: sólo tener en cuenta aquellas combinaciones interesantes de obstáculos. Esta tarea es totalmente dependiente del dominio sobre el cual se está trabajando.

1.3.2.4.3.Ejemplo

A continuación se muestra la jerarquía de objetivos encontrada para el problema del meeting scheduler.

Ilustración 17

A partir de este domino se pueden describir distintos sistemas, ellos diferirán en la forma en que los objetivos son operacionalizados. En la siguiente tabla se presentan los objetivos y las acciones a realizarse por el sistema, los participantes de la reunión y el iniciador (quien convoca a la reunión).

Objetivos	Acciones a realizarse
Scheduler disponible	Iniciador invoca scheduler
Pedido de restricciones de participantes	Iniciador especifica participantes Iniciador especifica detalles de reunión Scheduler compone invitaciones Subsistema de mail envía invitaciones
Restricciones de participantes	Participante compone mensaje de restricciones Subsistema de mail envía mensajes de restricciones
Establecer fecha sin negociación	Scheduler determina fecha sin negociación posible en base a restricciones de participantes
Preferencias conocidas de	Scheduler determina fechas posibles Scheduler compone pedidos de las preferencias de fecha

participantes	Subsistema de mail envía los pedidos Participante compone respuesta con las fechas preferidas Subsistema de mail envía las respuestas
Aprobación final	Scheduler determina fecha de reunión Scheduler compone mensaje de pedido de aprobación Subsistema de mail envía pedidos de aprobación Iniciador compone mensaje de aprobación
Participantes notificados	Scheduler compone notificación Subsistema de mail envía notificaciones

Ilustración 18

A partir de esta operacionalización de objetivos se pueden formar la tabla con los objetivos, obstáculos y formas de evitar dichos obstáculos. Sólo son tenidos en cuenta los objetivos funcionales.

Objetivo	Obstáculo	Formas de evitar/mitigar el obstáculo
Scheduler disponible	**El iniciador se olvida como invocar al scheduler**	**Nada**
	El scheduler falla	**Se vuelve a invocar al scheduler** **Existe un subsistema backup disponible**
Pedido de restricciones de participantes	**Se invita gente que no tenía que invitarse**	**Avisar a los participantes incorrectos** **Se amplían las fechas posibles** **El iniciador confirma detalles**
	Se dan detalles equivocados de la reunión	**Se piden las restricciones de los participantes.**
	No se envían las invitaciones	**Se piden las restricciones de los participantes.**
Se provee las restricciones de los participantes	**Los participantes ignoran el pedido de restricciones**	**Se piden las restricciones de los participantes** **No se considera a esos participantes para la selección de fecha**
	Los mensajes de restricciones de los participantes no se envían	**Se piden restricciones de los participantes** **No se considera a esos participantes para la selección de fecha**
Planeamiento de la reunión	**No hay fecha posible**	**Se amplían los rangos de fechas**
	Los participantes cambian sus fechas	**No se consideran a esos participantes para la selección de fecha**
Conocer las	**No se envían las**	**Se fija la fecha de reunión sin**

preferencias de los participantes	**preferencias**	**negociaciones** **Se piden las preferencias de los participantes**
	Los participantes ignoran el pedido de preferencias	**Se piden las preferencias de los participantes** **Se asumen las preferencias de los participantes**
	Los mensajes de preferencias no son enviados	**Se piden las preferencias de los participantes** **Se asumen las preferencias de los participantes**
Pedido de aprobación final	**No existe una fecha final**	**Se fija la fecha de reunión sin negociación** **Los participantes amplían sus fechas**
	No se envía el pedido de aprobación	**Se requiere nuevamente el pedido de aprobación**
	El iniciador ignora el pedido de aprobación	**Se requiere nuevamente el pedido de aprobación** **Se fija la fecha de reunión sin negociación**
Notificación de los participantes	**No se envía la notificación**	**Participantes notificados**
	Los participantes ignoran la notificación	**Nada**

Ilustración 19

Combinando los obstáculos con los objetivos que impiden, se encuentran los casos que pueden ser investigados a través de escenarios.

El siguiente es un ejemplo de un escenario escrito en forma textual, sin seguir la estructura de setting y narrativa propuesto anteriormente.

> *Escenario: se invito a la reunión a gente equivocada, se amplían las posibles fechas de reunión.*
>
> *Colin quiere realizar una reunión esta semana. Invoca al scheduler y le da la lista de invitados en donde incluye por error a Idris y Annie. Todos los invitados responden y el scheduler propone como posible fecha el miércoles de 15 a 16 hs o el jueves de 14 a 15 hs. Cuando Colin se da cuenta del error cometido, saca de las listas a Annie e Idris, con lo cual el scheduler expande las posibles fechas, incluyendo el miércoles de 11 a 12 hs. Envía las posibles fechas a los participantes correctos y mensajes de disculpa a Annie e Idris. El scheduler selecciona miércoles de 11 a 12 a partir de las preferencias de los participantes y le comunica a Colin. Este aprueba la fecha y se envían a los participantes la notificación.*

1.3.2.4.4.Mejoras de la nueva propuesta

Es interesante ver estos dos trabajos del mismo autor, ya que se puede apreciar la inclusión de reglas para el desarrollo de escenarios en [Potts 95] que permiten que la construcción sea en forma sistemática.

En [Potts 94], en donde los escenarios surgían de forma espontánea, se encontraron 14 escenarios. En [Potts 95], teniendo en cuenta que sólo se consideró un obstáculo en cada escenario ignorando posibles combinaciones, se han encontrado 31 escenarios.

De los 31 escenarios sólo 5 fueron encontrados en los 14 iniciales. Los 9 restantes sólo consideran las mismas situaciones pero con más detalle. Los 26 escenarios nuevos no eran cubiertos en el primer estudio.

1.3.2.5.Aries

La representación elegida para los escenarios es principalmente textual (con algunos gráficos que permite una mejor compresión). Cada escenario está compuesto por dos partes: situación y comportamiento. Ambos están formados por un conjunto de estados ordenados según alguna relación (secuencia, concurrencia, repetición, inclusión negación, exclusión, etc). La situación describe el estado inicial del sistema y los estados que va tomando a medida que el escenario transcurre (las invariantes). El comportamiento describe lo que hace el sistema teniendo en cuenta relaciones entre estados que pueden estimular, permitir o inhabilitar otro estado.

Por ejemplo, suponiendo un escenario que describe un cruce de calles con semáforos. Si se quiere analizar el cruce de autos, el comportamiento estará dado por: auto frente al cruce, auto dentro el cruce, auto después del cruce. Y la situación sería el estado de las luces del semáforo, como van variando a lo largo del escenario. Si se desea analizar como responden las luces a la presencia o ausencia de luces (suponiendo que así lo hicieran) el conocimiento es reorganizado y las luces pasan a ser parte del comportamiento y los estados del auto la situación. Cuando se usan los escenarios para investigar sobre la funcionalidad del sistema se esta probando la situación o el comportamiento: que comportamiento existe ante determinada situación o que situación tiene que haber para que un determinado comportamiento ocurra.

Los escenarios son usados en esta metodología para: describir y clarificar propiedades relevantes del dominio de aplicación, encontrar requerimientos, evaluar alternativas de diseño y validar el diseño.

Se implementó una herramienta **aries**, a través de la cual se pueden grabar escenarios (en un modelo hypertextual), ejecutarlos y visualizarlos. Esta herramienta consta de tres partes principales:

rad: es un sistema mediante el cual los expertos en el dominio de la aplicación pueden describir sus propios escenarios, definiendo los objetos y asignándole comportamiento para las distintas situaciones. El aporte novedoso es que mantiene separado el comportamiento de la situación. Existe un ambiente para implementar este método. A través de una interfase amigable, el usuario tiene varias alternativas para describir un escenario. Los mismos pueden ser ejecutados usando el simulador **paisley**. El sistema es usado para análisis, evaluar alternativas de diseño y validar diseños.

asc: es una herramienta que permite simulación para que el usuario pueda investigar sobre la funcionalidad del sistema.

help: es el sistema de ayuda del sistema que emplea escenarios de entrenamiento para los usuarios del sistema. Describe el conjunto de acciones que se debe realizar para determinada tarea y crea el entorno apropiado para que el usuario realice la misma tarea.

1.3.2.6.El modelo de Carroll

Este autor proviene del área **hci**[2] por lo cual considera un escenario como el uso (real o imaginario) que tiene el usuario de un sistema, analizando el comportamiento deseado o no deseado del individuo frente al mismo.

Un escenario describe en forma textual una situación particular de un usuario interactuando con el sistema. Esta información sirve como fuente para el diseño del sistema. A través del escenario se puede observar que hace el usuario con el sistema, como interactua, como reacciona ante las respuestas y que problemas tiene. El conjunto de escenarios permite razonar sobre el comportamiento de los usuarios ante determinadas situaciones del sistema.

La creación de escenarios no puede basarse en la simple observación del sistema. A veces se puede necesitar escenarios de sistemas que aún no existen o del cual no se conoce su uso. Esto se puede hacer por analogía con sistemas ya existentes (el nuevo sistema puede ser similar a otro ya existente o ser descendiente de otro sistema).

Si bien mencionan el uso de escenarios para la etapa de adquisición de requerimientos, su trabajo esta enfocado en el uso de escenarios para diseño. El conjunto de escenarios muestra como actúan los usuarios ante determinadas situaciones y sobre esta base se pueden discutir alternativas de diseño. Propone el diseño racional, a partir de un escenario analizar las relaciones causales del mismo: ante determinada situación se puede desencadenar una reacción favorable o no en el usuarios. Cada situación del escenario debe ser analizada de la siguiente forma:

> *En <situación> <una expresión> causa <consecuencias deseables> pero puede causar <consecuencias indeseables>*

A partir de aquí se pueden analizar escenarios alternativos en donde distintas condiciones del sistema intentan obviar las consecuencias no deseadas.

1.3.2.7.La propuesta de Booch

¿Para que sirve un escenario en el desarrollo de soft orientado a objetos? Según [Booch 94], cumplen tres principios fundamentales. En primer lugar los escenarios son una parte esencial para capturar los requerimientos. Los escenarios hablan el lenguaje del usuario final y del experto del domino, por lo tanto proveen un medio para que ellos expliquen sus expectativas sobre el comportamiento del sistema. Segundo, los escenarios proveen un vehículo de comunicación. Llevan al usuario final y experto del dominio al nivel del problema, exigiendo al desarrollador a empaparse en el dominio del problema, forzándolo a considerar una distribución inteligente de responsabilidades dentro del

[2] Human Computer Interfase

sistema. Y tercero, a medida que el proyecto avanza, los escenarios sirven como instrucciones a los desarrolladores individuales tanto, como al equipo de pruebas.

¿Qué es un escenario? Un escenario provee un esbozo del comportamiento del sistema. Los escenarios documentan decisiones de requerimientos o diseño, proveen un punto de comunicación sobre la semántica del sistema y pueden servir como partida para la implementación detallada.

Aún los sistemas más complejos pueden ser caracterizados en función de una docena de escenarios. Esto es verdad todavía en dominios tan sofisticados como telefonía o control de trafico aéreo. En cada caso, sólo hay un modesto número de escenarios primarios que modelan el problema central. Básicamente, un escenario primario representa alguna función fundamental del sistema. Por ejemplo, un escenario primario de un sistema telefónico, involucraría hacer una conexión de un teléfono a otro. En el sistema de control de trafico aéreo, el escenario de una aeronave entrando al aeroespacio controlado representa un comportamiento primario. Obviamente, aún el sistema más pequeño implica una multitud de posibles condiciones excepcionales y caminos alternativos. Estos escenarios son llamados escenarios secundarios. Se llaman así, no porque son de importancia secundaria, sino porque cada escenario secundario típicamente representa alguna variación sobre un escenario primario. De este modo, los escenarios primarios y secundarios forman una jerarquía: un escenario secundario es "una clase de" un escenario primario.

Luego, el comportamiento del sistema de software puede ser capturado a través de una red de escenarios de la misma forma que lo hace los stoyboards con respecto a una película. Aunque, la analogía entre el proceso de desarrollo de soft y filmar una película se quiebra en el punto que las películas tienen sólo un camino de acción y pueden ser representadas por un largo y continuo storyboard. En cambio, todos las aplicaciones de soft interesantes raramente tienen un camino simple de comportamiento.

¿Como se hace para representar los escenarios? Bien, Booch utiliza distintos métodos. En primer lugar usa tarjetas **CRC**. El probó que son una buena manera de abordar la construcción de escenario. Su mayor atractivo como técnica de desarrollo es que son totalmente libres. Lamentablemente, las tarjetas CRC padecen de una gran limitación, no pueden considerar aspectos temporales de un escenario. Los diagramas de interacción solucionan este problema. Ellos claramente muestran el flujo de la actividad en el tiempo. Pero también adolecen de un problema, sólo pueden mostrar actividad de colaboración en una forma secuencial. Los diagramas de interacción no pueden por ellos mismos mostrar mensajes concurrentes, consideraciones de tiempo o hechos tales como que un objeto en un diagrama es en realidad un atributo de otro. Entonces aparecen los diagramas de objetos que lo solucionan.

1.3.2.8.Modelo práctico

El artículo presenta un trabajo realizado en una empresa en donde se usó escenarios para la etapa de adquisición y validación de requerimientos. Este artículo no propone ningún modelo nuevo de desarrollo. Usa prácticamente la metodología de Jacobson para descubrir los actores y derivar los escenarios. y luego desarrollar el modelo.

Su valor radica en que se desarrolló un sistema real en donde estuvieron involucrados los usuarios y los ingenieros. Su experiencia marcó que las notaciones

semiformales (por ejemplo diagramas de entidad relación) no se comprenden fácilmente. Proponen una descripción en lenguaje natural organizada de la siguiente forma.

1. precondición
2. flujo principal de eventos
   2.1. eventos normales
   2.2. eventos excepcionales
   2.3. eventos extendidos
3. post condiciones
4. eventos paralelos

1.3.2.9.Firesmith

Este autor ha desarrollado una herramienta de desarrollo de soft de cuarta generación, denominada **adm**. Esta herramienta permite construir soft orientado a objetos, distribuido y de tiempo real. Uno de los elementos de los que dispone adm es de escenarios.

Adm define un escenario como al uso de una o mas clases u objetos en colaboración. Por ejemplo una secuencia legal de mensajes.

De esta forma, los escenarios son abstracciones funcionales que típicamente involucran varios objetos o clases que pueden involucrar múltiples hilos de ejecución. A diferencia de los use cases de Jacobson, los escenarios no necesitan ser iniciados por un actor. Por ejemplo, un loop de control en una aplicación se ejecuta automáticamente sin la iniciación de ningún cliente.

Los escenarios comúnmente son utilizados para: recoger requerimientos, documentar los requerimientos, comunicarse con los clientes, preparar integración y pruebas funcionales, entre otras cosas. Adm les da diferentes usos. Inicialmente los escenarios son usados para capturar el comportamiento general del soft desde el punto de vista del cliente. Luego se los utiliza para capturar mecanismos complejos en subsistemas. Y por último pueden ser utilizados para mostrar las interacciones entre las clases.

1.3.2.9.1.Modelando el ciclo de vida de los escenarios

En la mayoría de la aplicaciones, el orden de sus escenarios no es arbitrario. Uno o más escenarios iniciales son necesarios antes de que pueda ser ejecutado algún otro. Similarmente uno o más escenarios finales pueden terminar la aplicación. Algunos escenarios pueden ser ejecutados cuando cierta condición se cumple o una condición puede determinar cual de dos o más escenarios se puede ejecutar. Otros escenarios pueden ejecutarse un número de veces basado en una condición. Muchos escenarios a veces se ejecutan concurrentemente y deben estar sincronizados. Otras veces, sólo uno de varios escenarios que pueden ejecutarse debe realizarse. El ciclo de vida de los escenarios es la especificación del orden de los escenarios del sistema. A continuación se muestran las figuras que pueden formar un diagrama del ciclo de vida de los escenarios.

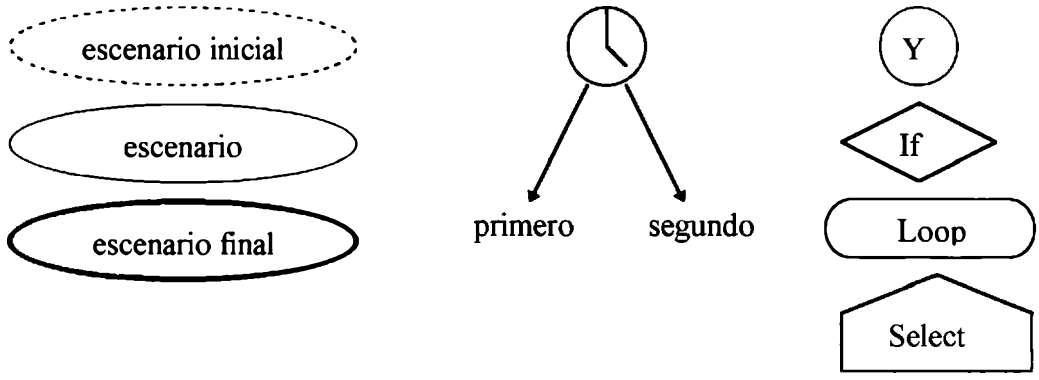


Ilustración 20

1.3.2.9.2.Modelando escenarios individuales

La especificación de un escenario se adecua al siguiente formato:

scenario scenario_identifier
definition is
abstraction
<statement of abstraction>
end abstraction;
responsabilities
<list of responsabilities>
end responsabilities;
end definition;
scenario scenario_identifier
interface is
protocol
<list of trigger>
<list of exception>
end protocol;
end interface;
scenario scenario_identifier
implementation is
invariants
<list of invariantes>
preconditions
<list of preconditions>
end preconditions;
body
<sequence of statements>
end body;
postconditions
<list of postconditions>
end postcontions;
end implementations;

1.3.2.10.Comparación de los distintos modelos

1.3.2.10.1.Representación de escenarios

De todas las definiciones de escenarios, una de las más completas está dada en el modelo **icm**. En él, se tienen en cuenta los actores, sus objetivos, las acciones que realizan para alcanzarlo y los posibles obstáculos que impiden esos objetivos. Es uno de los mejores enfoques para describir el uso particular de un sistema: cuando se está haciendo uso de un sistema es porque determinadas entidades precisan satisfacer alguna necesidad (objetivo) y lo hacen a través de distintas acciones, las cuales no siempre se pueden realizar, por distintos obstáculos provocados por acciones erróneas de las entidades o por estados que ha alcanzado el sistema.

Un elemento que aparece en casi todos los trabajos, son las pre y post condiciones. Las mismas, dan una idea de estado inicial y final de un escenario, como se hace en el modelo de **oba**. Los distintos trabajos, si bien lo denominan de otra forma, también tienen presente esta idea. En icm, la postcondición sería el outcome del último episodio. En aries, sería el último estado de la descripción de la situación. En cuanto a las precondiciones, sucede lo mismo, en todos los trabajos, son considerados estados iniciales en los cuales los escenarios comienzan.

Con respecto a los actores, si bien en todas las representaciones se mencionan, en el único trabajo que se identifican a estos como entidades externas del sistema es en el modelo de use cases. Esto significa que es necesario conocer los límites del sistema, los que en realidad se están tratando de definir. No siempre es fácil identificar en un primer momento que entidades son externas y cuales internas al sistema.

Del modelo práctico presentado, se concluye, que si bien cuanto más estructurada es la representación de escenarios, es mas fácil la compresión y seguimiento, la notación no debe ser dificil de entender. Los escenarios son una herramienta de comunicación entre los expertos del sistema y los del dominio, y a estos últimos les resulta dificil utilizar notación específicas. Es preferible descripciones textuales en lenguaje natural.

1.3.2.10.2.Relaciones entre escenarios

En este punto se tienen que distinguir entre dos tipos de relaciones.

El primer tipo de relación es el que se menciona en use cases. El objetivo de las relaciones usa y extiende es poder expresar la semántica de un escenario en función de otros. Con ellas se evitan redundancia de información y sólo se debe especificar comportamiento novedoso. Son relaciones de diseño e intentan mejorar la semántica de cada escenario mediante abstracciones.

El segundo tipo de relación es el que se trata en el modelo de Firesmith. Lo que hace es organizar los distintos escenarios para poder interpretar el comportamiento global del sistema. Se utilizan estructuras de control como las que están en cualquier lenguaje de programación imperativo: secuencia, iteración y bifurcación para integrar a todo el conjunto de escenarios. Estas son relaciones de dominio, ya que los lazos entre los escenarios están dados por el propio dominio estudiado.

1.3.2.10.3.Uso de los escenarios en el desarrollo del software

En general el uso de escenarios es para la etapa de análisis de requerimientos. Comenzando con técnicas tradicionales como ser entrevistas, observación y análisis de documentación existente, se derivan los primeros escenarios. Luego estos escenarios son discutidos con los usuarios para encontrar situaciones no consideradas y proponer alternativas de diseño.

Por lo general el uso de escenarios se puede hacer bajo cualquier metodología de desarrollo, siendo un complemento con una visión orientada al usuario. Sin embargo, es posible integrar los escenarios al proceso de desarrollo, como lo hace Jacobson en su trabajo.

1.3.3.Estructura de los escenarios

En esta sección se describe la estructura que se adopta en este trabajo para representar los escenarios. El diagrama entidad relación aparece en la página 13, Ilustración 2. Se recordará que las entidades que lo conforman son: nombre, objetivo, actores, recursos, contexto y episodios. Todas las entidades se expresan con texto en lenguaje natural que guarda un cierto formato, salvo, como el diagrama lo muestra, los episodios que pueden ser a vez otros escenarios.

En este punto se describen cada una de estas entidades:

- Titulo: Es el título del escenario. En el caso de un subescenario, como se ve más adelante, el título es el mismo que la oración del episodio sin las excepciones y/o restricciones.

  [frase | ([actor | recurso] + verbo + predicado)]

  Ejemplo: Solicitar un nuevo pasaporte (frase)

- Objetivo: Es la meta que debe ser alcanzada en el sistema. El escenario describe el logro del objetivo. Es descripto por la siguiente estructura

  [sujeto] + verbo + predicado

  Ejemplo: Cobrar (verbo) al ciudadano el impuesto (predicado)

- Contexto: Describe la ubicación geográfica y temporal del escenario y algún estado inicial importante. El contexto es representado por oraciones con el siguiente formato:

  Ubicación + Estado
  Donde ubicación es: nombre
  Y estado es :[actor | recurso] + verbo + { restricciones }

  Ejemplo: En la caja (nombre). El ciudadano (actor) ha recibido (verbo) los formularios (predicado) que deben coincidir con los del cajero (restricciones)

- Recurso: Soportes, dispositivos necesarios que deben estar disponibles en el escenario. Describen entidades pasivas con las cuales operan los actores.

  nombre + { restricciones }

  Ejemplo: Talonario (nombre),el cual debe contener el nombre de los ciudadanos que solicitan pasaporte (restricción).

- Actor: Es una persona o entidad que juega un rol en el escenario. Se lo representa:

  nombre

  Ejemplo: Cajero (nombre)

- Episodios: Cada episodio representa una acción realizada por un actor donde participan otros actores y se usan recursos. Es una serie de oraciones que describe el escenario y muestra su comportamiento. La siguiente gramática BNF muestra como se deben estructurar los episodios:

  <episodios>::= <serie>
  <serie>::= <oración> | <serie> <oración>
  <oración> ::= <oración secuencial> | <oración no secuencial> | <oración condicional>
  <oración secuencial> ::=<oracion de episodio> **CR**
  <oración condicional> ::=**if** <condición> **then** <oración de episodio> **CR**
  <oración no secuencial> ::= # <serie> #

  Donde <oración de episodio> se la describe de la siguiente forma:

  [actor | recurso] + verbo + predicado + {restricción} + {excepción}

  Ejemplo: El fotografo (actor) toma (verbo) la fotografia del ciudadano (predicado). Exception: que la cámara no funcione.

Los atributos más importantes del modelo ER son las restricciones y las excepciones. Una restricción es el alcance referido a una entidad dada. Es representado por una oración con la siguiente estructura: **debe**+verbo+predicado. Una excepción causa serios quiebres en los escenarios, originando un conjunto de acciones diferente, descritos en forma separada como escenarios de excepciones. Es representado por una oración o por un párrafo pequeño y generalmente refleja la falta o malfuncionamiento de un recurso necesario.

Los términos: nombre, ubicación, sujeto, verbo, predicado, actor y recurso pueden ser elegidos desde el LEL, haciendo posible el uso de un vocabulario controlado.

A continuación se da el ejemplo de un escenario perteneciente al trabajo de la emisión de pasaportes:

titulo: solicitar un nuevo pasaporte.
objetivo: satisfacer los requerimientos iniciales para un nuevo pasaporte.
contexto: división de certificación y documentos. El ciudadano no tiene su pasaporte.
actores: ciudadano, cajero, responsable de tomar las huellas dactiloscópicas, fotógrafo, responsable de la oficina de certificación y documentos.

recursos: cámara, formulario, pasaporte en blanco, talonario, documentos del ciudadano.

episodios:

El ciudadano llena el formulario. Restriction: Llenarlo con tinta.

El responsable de la oficina de documentos y certificados revisa los formularios. Restriction: El formulario debe estar llenado correctamente . Exception: Documentos faltantes.

El fotógrafo toma una fotografía del ciudadano. Exception: La cámara no funciona.

El responsable de la oficina de huellas toma las muestras dactiloscópicas del ciudadano #

El ciudadano paga los impuestos del pasaporte

El ciudadano firma el pasaporte.

El ciudadano recibe su comprobante.

1.3.4. Vistas de los escenarios

En la sección 1.2.2, al analizar la estructura del LEL, se ve que tiene un aspecto de hypertexto. Desde cada nodo, que está compuesto por un signo a definir y su descripción, es posible navegar a otros términos, los cuales se utilizan para definir al primero. En esta sección, se verá que los escenarios también se pueden organizar en un hypertexto. Básicamente un escenario puede estar ligado con otro porque el segundo es un subescenario del primero (forma parte de un episodio), porque contiene un mismo signo del LEL, o porque se lo defina explícitamente por parte del usuario. Además de esta vista hypertextual, se tiene la vista configuración. Como el modelo de escenarios evoluciona y cambia constantemente, es necesario organizar el modelo en versiones. Cada versión contiene el modelo de escenarios en un momento dado, como así también las razones por las cuales se modificó el modelo anterior y se llegó al actual. A continuación se ven cada una de estas vistas.

1.3.4.1. Vista hypertextual

Cada escenario representa un nodo en el hypertexto y los links pueden obtenerse de tres formas:

- de las relaciones estructurales entre escenarios, es decir de la relación subescenario.
- de la información del LEL.
- definidos por el usuario.

Estos tres tipos de links definen tres contextos navegacionales. El primer contexto navegacional está formado por todos los subescenarios de uno dado. Para este contexto se tiene que brindar la posibilidad de alcanzar el primer subescenario y cada uno de los siguientes.

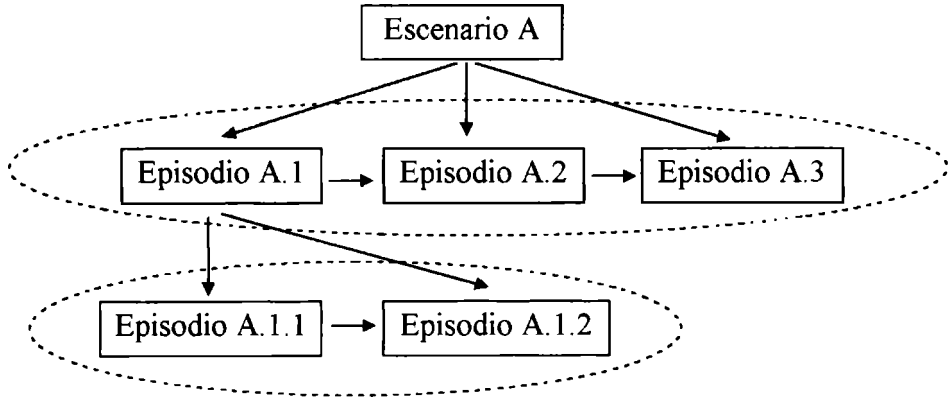


Ilustración 21. Contexto navegacional

Como puede verse en la Ilustración 21, se han agregado también links entre los componentes del escenario A. Cuando se está navegando en este contexto los episodios de A, se entiende que todos los componentes están conectados por un link **next**, inducido por la composición estructural de A. Lo mismo sucede con las episodios de A.1.

Otro contexto interesante es derivado de la información contenida en el LEL. Por ejemplo se podría seleccionar una entrada del LEL representando un recurso en el modelo de escenarios y buscar todos los escenarios que utilicen ese recurso. Estos nodos podrían no tener una relación jerárquica como la del punto anterior, sin embargo se organizarían de la misma forma que los episodios, se tendría un conjunto de escenarios que pueden ser accedidos desde un término y para cada escenario habría un next.

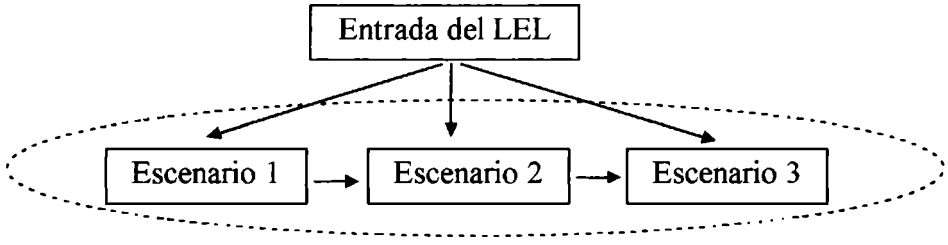


Ilustración 22

Además de estos dos casos en donde los escenarios poseen una relación estructural o comparten términos del LEL, podría darse la necesidad de querer linkear dos escenarios, los cuales no se ajusten a ninguna de las opciones anteriores. Esta clase de links debe ser descripto explícitamente por el usuario. El manejo es similar al anterior, una vez identificado este tercer contexto navegacional, de cada escenario se puede ir directamente a los escenarios definidos y moverse por este conjunto con la operación siguiente.

Además de los tres tipos de links definidos entre los escenarios, existe otro tipo de links que va desde el modelo de escenarios hacia el modelo del LEL. Los anchors los forman todos los términos utilizados en la descripción de los escenarios que se hayan definido en el LEL. De esta manera, clickeando sobre la palabra o frase marcada es posible ver su definición.

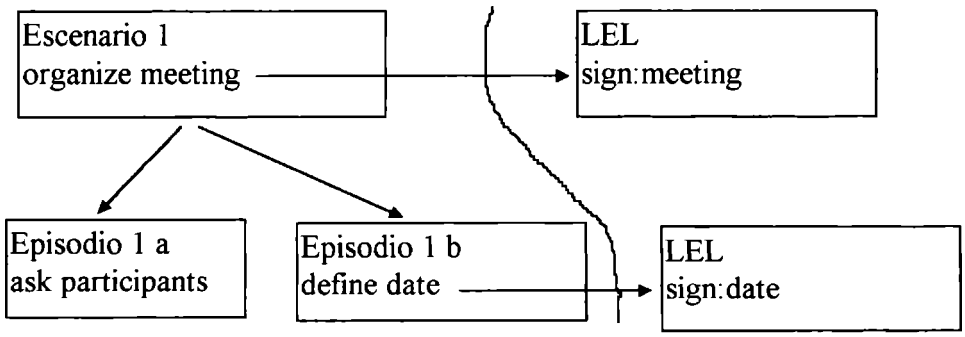


Ilustración 23. Links entre los modelos de escenarios y LEL

1.3.4.2. Vista de configuración

Como se comenta en secciones previas, en función de la información del LEL y los escenarios, se obtiene un modelo preliminar de objetos. Como el LEL y los escenarios evolucionan mientras se lleva a cabo el diseño, es necesario saber como era exactamente el modelo de LEL y escenarios al tiempo de construirse cierto modelo preliminar. Es por esto que aparecen las vistas de configuración. Cada configuración representa un modelo que difiere del anterior por algún cambio en la definición de algún escenario. Estos cambios pueden ser alteración de información o el agregado de nueva. Cada versión de cada modelo mantiene la siguiente información: fecha, hora, persona que hace el cambio, razón del cambio y tipo (agregado, modificación o exclusión).

A continuación, a manera de ejemplo se brinda la versión 1, de un escenario correspondiente al trabajo de emisión de pasaportes.

- *Fecha: 15/2/96*
- *Hora: 14:00 hs*
- *Usuario: Federico*
- *Causante: Caso de estudios de los pasaportes*
- *Fecha de la causa: 20/10/95*
- *Tipo: Inclusión*

Titulo:

El responsable de la oficina de huellas dactilares toma las impresiones dactiloscópicas del ciudadano.

Objetivo:

Obtener la identificación dactiloscópica del ciudadano.

Contexto:

Oficina de dactiloscopia. El ciudadano tiene el formulario autorizado.

Actores:

Responsable de la oficina de huellas dactilares.
Ciudadano.

Recursos:

Formulario, que debe ser previamente chequeado.
Tinta.
Formulario de dactiloscopia, el cual debe ser presentado si el ciudadano no tiene otra identificación.

Episodes:

El ciudadano le da el formulario al responsable de la oficina.
Si el ciudadano no tiene identificación previa, el responsable de la oficina imprime las huellas del ciudadano en el formulario.
El responsable de la oficina imprime el pulgar derecho del ciudadano en el formulario.
El ciudadano se limpia su mano en una toalla de papel.
El responsable de la oficina le devuelve el formulario al ciudadano.
Si el ciudadano no tiene identificación previa, el responsable de la oficina le devuelve al ciudadano el formulario.

A continuación, la versión número 2, del mismo escenario.

- *Fecha: 15/3/96*

- *Hora: 15:30 hs*
- *Usuario: Federico*
- *Causante: Discusión sobre como el responsable de la oficina trabajaría con una maquina lectora de huellas dactilares.*
- *Fecha de la causa: 1/3/96*
- *Tipo: Modificación*

Titulo:

El responsable de la oficina de huellas dactilares toma las impresiones dactiloscópicas del ciudadano.

Objetivo:

Obtener la identificación dactiloscópica del ciudadano.

Contexto:

Oficina de dactiloscopia. El ciudadano tiene el formulario autorizado.

Actores:

Responsable de la oficina de huellas dactilares.
Ciudadano.

Recursos:

Formulario, que debe ser previamente chequeado.
Máquina lectora de huellas dactilares.
Base de datos de huellas dactilares.

Episodes:

El ciudadano le da el formulario al responsable de la oficina.
El responsable de la oficina le da instrucciones de como posicionar la mano al ciudadano.
El responsable de la oficina posiciona el formulario para imprimir y enciende la maquina.
Si la señal de la maquina lectora de huellas dactilares es roja, el responsable de la oficina se asegura de la los dedos del ciudadano estén correctamente posicionados y reenciende la maquina.
 # La máquina imprime las huellas en el formulario.
La máquina graba la información en la base de datos. Restriccion: La grabación se debe realizar en menos de 4 segundos.
#

1.4.Clases candidatas

Como resultado de la etapa de análisis de requerimientos se necesita reflejar de la forma mas exacta posible el problema y restricciones de las posibles soluciones.

1.4.1.Modelo preliminar de objetos

Los pasos que se deben seguir para derivar el modelo orientado a objetos son los siguientes:

- Hallar clases primarias y sus responsabilidades.
- Hallar clases secundarias y sus responsabilidades.
- Hallar y refinar colaboraciones y responsabilidades.
- Validar por medio de tarjetas CRC
- Definir el modelo lógico

1.4.1.1.Hallar clases primarias y sus responsabilidades

Las clases primarias se caracterizan por representar un elemento activo dentro del macrosistema y poseer un comportamiento importante. Los actores de los escenarios responden a estas características. Por lo tanto, para hallar las clases primarias, es suficiente con tomar cada uno de los actores de los escenarios. Una vez que se obtienen el conjunto de clases primarias, se deben definir sus responsabilidades.

Estos actores son elementos centrales del macrosistema, así que tendrán seguramente una entrada en el LEL, y de allí se tomarán las responsabilidades. Cada entrada en el LEL consta de noción e impacto. La noción es una descripción tradicional. Pero, los impactos representan la influencia que tiene el elemento entre los demás o, los demás en éste. Esta influencia no es otra cosa que los servicios que requiere de otras clases o los que provee. Así, que las responsabilidades se deben extraer de los impactos definidos en el LEL. Sin embargo, algunas distinciones deben ser hechas.

Puede suceder que el actor carezca de entrada en el LEL o, tenga entrada pero no posea impactos. Si se presenta esta situación, una revisión del LEL y escenario debe realizarse. Un hecho de este tipo nunca tendría que ocurrir, ya que el actor de un escenario es un elemento importante como para que se lo defina en el LEL y tiene como impactos las acciones que realiza en los episodios.

Por otro lado, si los impactos representan restricciones, se debe determinar el comportamiento necesario para que pueda completar sus responsabilidades.

Con la lista de clases primarias completa, se debe definir la jerarquía de herencia. Para tal fin se podrá valer de la noción del LEL. Esta jerarquía no debe expresar decisiones de diseño, debe representar el universo del discurso.

1.4.1.2.Hallar clases secundarias y sus responsabilidades

Las clases secundarias constituyen el soporte de las clases primarias para que puedan completar su funcionalidad. Por lo general representan objetos pasivos, que son repositorios de datos, que prestan colaboración a las clases primarias. En los escenarios cumplen el papel de recursos.

El método para encontrarlas, consiste en analizar las responsabilidades de las clases primarias, y seleccionar aquellas palabras que aparezcan en el LEL. Esto no significa que ya sean clases secundarias, se deben revisar los siguiente puntos:

•Si el símbolo no tiene impactos relevantes, se lo debe modelar como un atributo de la clase en la cual aparece. En próximas etapas, se lo podría modelar como un objeto, pero en esta etapa no se justifica.

•Si el símbolo tiene impactos relevantes, se lo debe modelar como clase secundaria (siendo estos impactos sus responsabilidades), ya que es necesitada por la clase primaria para cumplir sus servicios.

•Si el símbolo representa una acción, se debe analizar si la acción tiene características que justifiquen que sea modelada como una clase (un ejemplo común son transacciones como depósitos, extracciones). En caso contrario, debe ser modelada como una responsabilidad de la clase involucrada.

1.4.1.3.Buscando y refinando colaboraciones y responsabilidades

En los dos apartados anteriores se detalla como hallar clases primarias y secundarias, con sus responsabilidades. En esta etapa se trata como refinar las responsabilidades y colaboraciones con el fin de poder armar las tarjetas CRC, para confrontarlas contra el experto del dominio.

Para cada clase (ya sea primaria o secundaria), se deben analizar los episodios de los escenarios en los que aparece, refinando sus responsabilidades con las acciones concretas que la clase debe hacer para cumplir con sus servicios.

En esta fase se deben considerar los siguientes puntos.

•En el momento en que se refinan las responsabilidades, también se definen colaboraciones, ya que se identifican las clases con las cuales colabora en las acciones de un servicio.

•Si algún símbolo del LEL no analizado aparece en los escenarios con un comportamiento relevante, se lo debe definir como clase. Mientras se refina una clase primaria, esto nunca puede suceder, ya que se analizaron los escenarios para buscar las clases secundarias. Pero puede suceder que una clase secundaria necesite de otra. Por lo que cabe la posibilidad de que aparezca una nueva clase como soporte de la secundaria.

•Nuevas responsabilidades pueden aparecer del acceso y mantenimiento del estado interno de una clase. Estas responsabilidades pueden no figurar en los impactos porque no son relevantes para el macrosistema, sin embargo se las debe agregar a las responsabilidades de la clase en cuestión.

Con esta información se está en condiciones de construir tarjetas CRC. Las mismas están compuestas de la siguiente forma:

Nombre: Justificación:		
Responsabilidades	Servicios	Contexto

Donde, nombre es el nombre de la clase. Justificación es la razón de incluirla como clase. Puede deberse a: ser actor en un escenario, poseer impacto relevante, ser el soporte de una clase primaria. Las responsabilidades son los impactos del LEL que se indican como refinar en esta sección. Los servicios son las clases con las que colabora y por medio de que protocolo. Y por último el contexto es de que escenarios se obtuvo los servicios respectivos.

1.4.1.4. Validación por medio de tarjetas CRC

Consiste en realizar una entrevista con los expertos del dominio, en donde el analista muestra por medio de las tarjetas las entidades que serán modeladas, con sus responsabilidades y colaboraciones. El modelo de LEL y escenarios se deben tener a mano para registrar cualquier cambio que surja de modificaciones en las tarjetas.

1.4.1.5. Definición del modelo lógico

Para cada tarjeta se debe definir un objeto siguiendo la técnica de modelo de objetos propuesta por Rumbaugh en [Rumbaugh 91]. Allí a cada objeto es representado por un recuadro en donde se detalla el nombre de clase, los atributos (variables) y los métodos. Luego se deben trazar los diagramas de asociación y los de interconexión.

2.Herramienta

Programar una computadora es una de las tareas mas difíciles jamas realizada por los seres humanos. Se requiere talento, creatividad, experiencia, la habilidad de construir y usar abstracciones y la disponibilidad de las mejores herramientas. (Timothy Budd)

2.1.Generalidades

La herramienta realiza chequeos de consistencia y revisiones que hacerlas en forma manual serían muy costosas. Deriva las tarjetas CRCs siguiendo el algoritmo que se ve en la primera parte. Sin embargo, su funcionalidad esta recortada con respecto a los modelos originales.

2.1.1.Motivación

La herramienta que se comenta en esta sección pretende automatizar y simplificar el uso del modelo presentado en la primera parte.

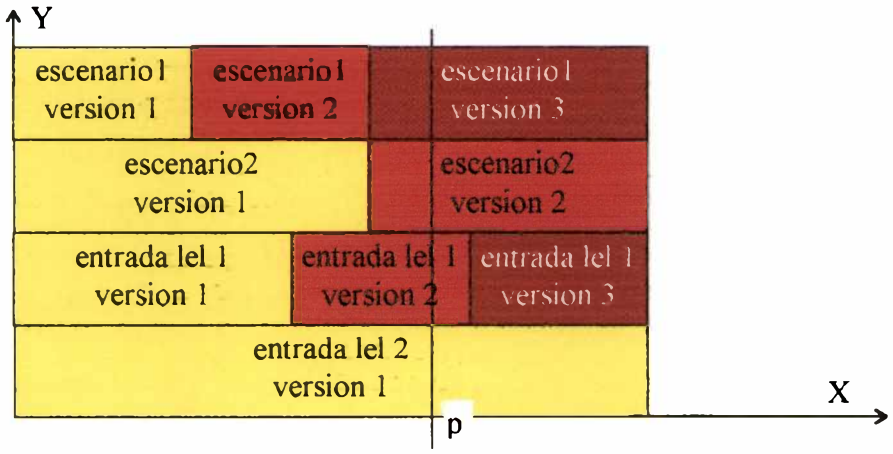
Entre las facilidades que ofrece sobresalen las siguientes:

- Maneja varios proyectos.
- Administra distintas versiones para cada proyecto.
- Controla que la evolución de las versiones sea solamente lineal.
- Al editar un escenarios o entrada del LEL, se puede hacer explícita la relación con otro elemento (link), antes de crear este último.
- Se dispone de las funciones cortar, copiar y pegar.
- Las sintaxis fija de oraciones (como la estructura IF...THEN... de los episodios de escenarios) se introduce por menú.
- Al finalizar la edición, se crean automáticamente links que no hayan sido definidos por parte del usuario.
- Si se modifica el titulo de un escenario o sinónimo de un LEL, que es link de otro elemento, se modifica el string del anchor.
- Identifica a los links de LELs, escenarios y CRCs de forma diferente, tanto durante la edición como la navegación.
- Mientras se navega, reconoce los contextos navegacionales, permitiendo ir al siguiente y anterior del contexto.
- Exporta la información de la version a formato html para que pueda ser recorrida por medio de navegadores de internet ordinarios.
- Chequea que no hayan títulos de escenarios repetidos o que varios LELs compartan algún sinónimo.
- Deriva en forma automática las tarjetas CRCs.

2.1.2.Restricciones de implementación

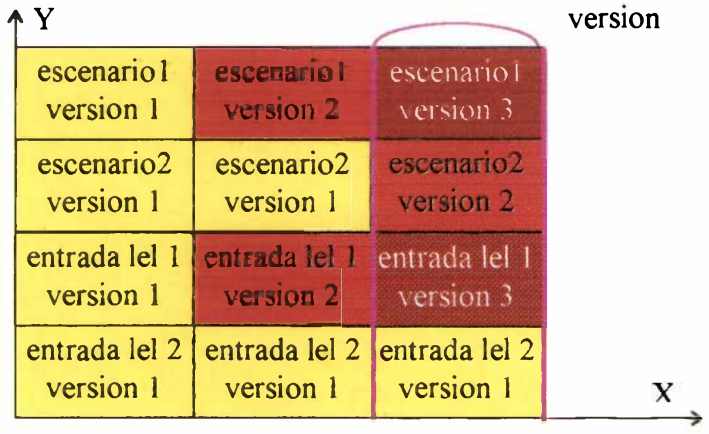
La herramienta no implementa completamente toda la funcionalidad necesaria para cumplir con los requerimientos expuestos en la primera mitad del trabajo. En esta primera versión de la herramienta se dejan de lado algunas funciones, las que se agregarán en posteriores implementaciones. Los siguientes puntos detallan las decisiones que se han tomado en la construcción de la herramienta:

- Tanto las entradas del LEL como los escenarios se describen en forma textual, y este texto tiene un formato específico, tal como se puede ver en "Estructura" en la página 18 y en "Estructura de los escenarios" en la página 56. Por ejemplo, si se pretende determinar los impactos de un término que es sujeto de una oración, se deben indicar las acciones (verbos) que realiza el sujeto. Si en cambio se trata del objetivo de un escenario el texto debe respetar el siguiente formato: [sujeto] + verbo + predicado. La herramienta no hace ningún parsing sobre el texto ingresado para validar la entrada.
- La metodología de diseño propuesta tiene como función derivar a partir de entradas del LEL y escenarios un modelo lógico del sistema. Sin embargo, la herramienta no llega hasta la definición del modelo, lo que hace es derivar las tarjetas CRC.
- Con respecto a la noción de configuración que se les asocia a los escenarios y entradas del LEL, se quiere dejar en claro la siguiente decisión de diseño que se ha adoptado para esta primera versión de la herramienta. Cada escenario y entrada del LEL poseen distintas configuraciones, estas dejan constancia de la evolución de cada elemento durante el análisis de requerimientos. Sean 2 escenarios y 2 entradas del LEL, que evolucionan de acuerdo al siguiente gráfico:

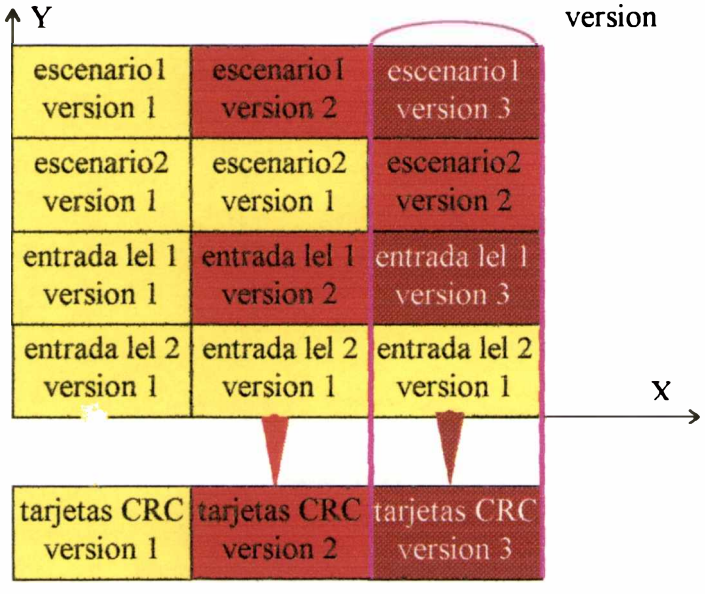


En un instante dado (que se identifica por un punto p en el eje x), se puede determinar cuales son las versiones del momento, trazando una recta perpendicular al eje X en p. Las bandas de configuraciones (versiones) con las que se interseque la recta determina las pertenecientes al instante p.

Esta herramienta no permite manejar configuraciones de la forma que se indica arriba, en donde una nueva configuración puede aparecer en cualquier instante de tiempo. Por el contrario, maneja una serie de eras (versiones), en donde cada cambio de era implica que se puede dar una nueva configuración.



Este manejo de versiones adoptado, simplifica la incorporación de las tarjetas CRC. La herramienta permite derivar las mismas en forma automática. La derivación se realiza a partir de los escenarios y las entradas del LEL. Pero como estos dos modelos evolucionan, se debe determinar de que configuraciones se derivan las tarjetas. Así que con la determinación de versiones, queda claro que las tarjetas se derivan de las configuraciones pertenecientes a la misma version.



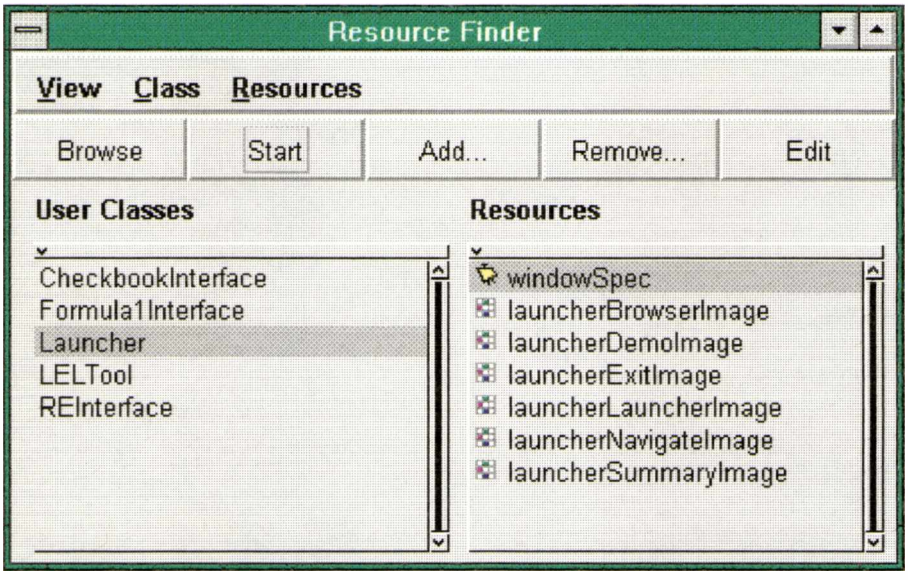
Para esta primera version de la herramienta, se prescinde de la información de configuración asociada a cada escenario y entrada del LEL (fecha, hora, usuario, causante, fecha de la causa y tipo: inclusión, modificación o exclusión). Puesto que cada version posee fecha y descripción, la fecha de la version funciona como cota de las fechas de las configuraciones y en la descripción se puede guardar información referente a usuarios y cambios relevantes con respecto a la version anterior.

2.2.Manual de uso

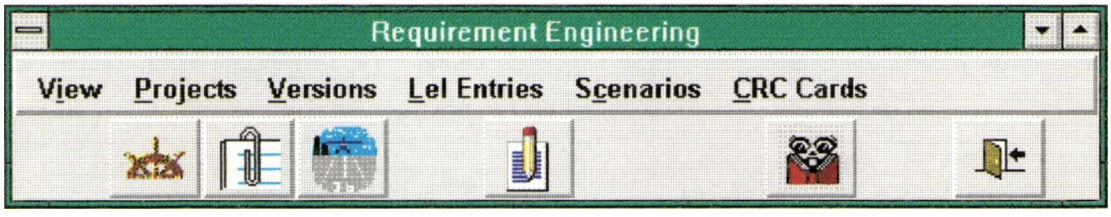
Si la industria automotriz, hubiera evolucionado en proporciones semejantes a las computadoras, hoy día un auto podría recorrer miles de kilómetros con un litro de combustible y alcanzar velocidades superiores a la del sonido, lamentablemente para poder abrir la puerta habría que leer un manual de doscientas paginas.

2.2.1.Arrancando la aplicación

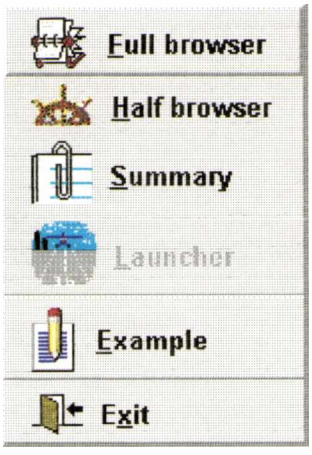
Desde el Resource Finder se debe elegir el recurso windowSpec de la clase Launcher y de esta forma se ejecuta la aplicación.



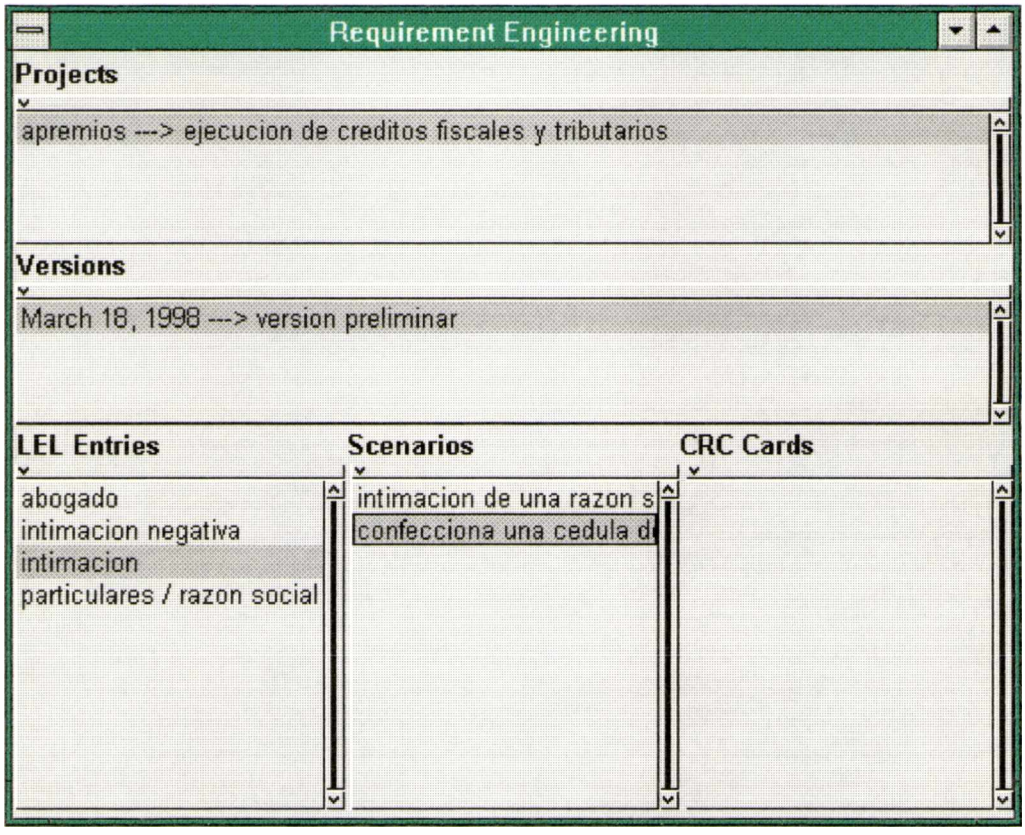
La ventana principal es la siguiente:



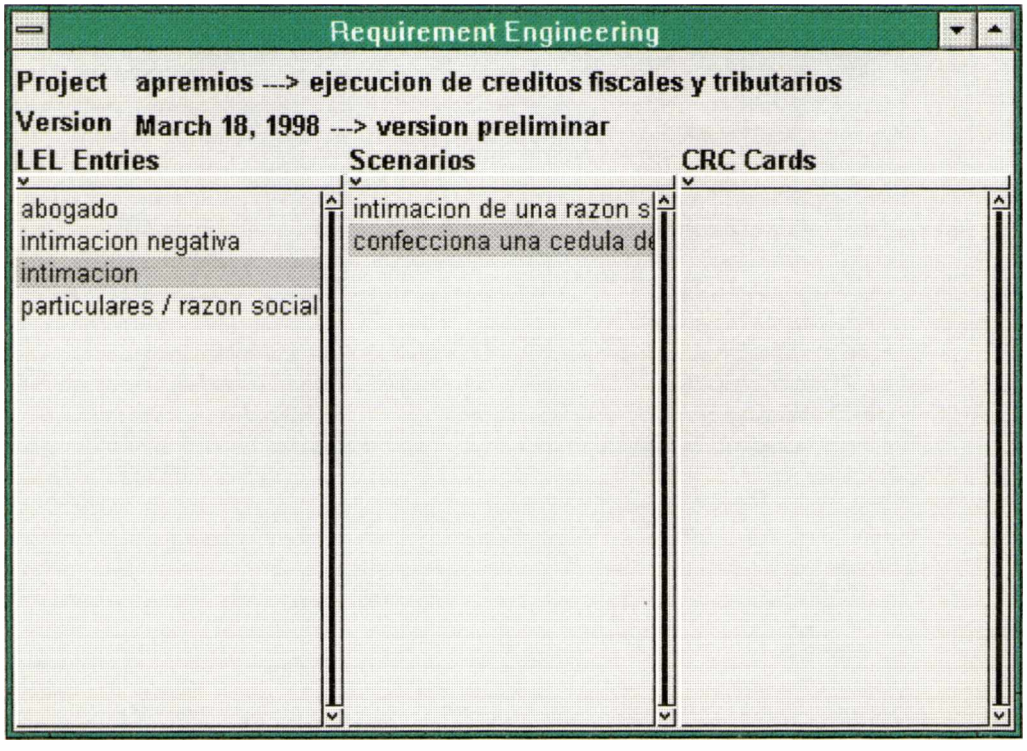
El menú que se despliega asociado a la opción View es el siguiente:



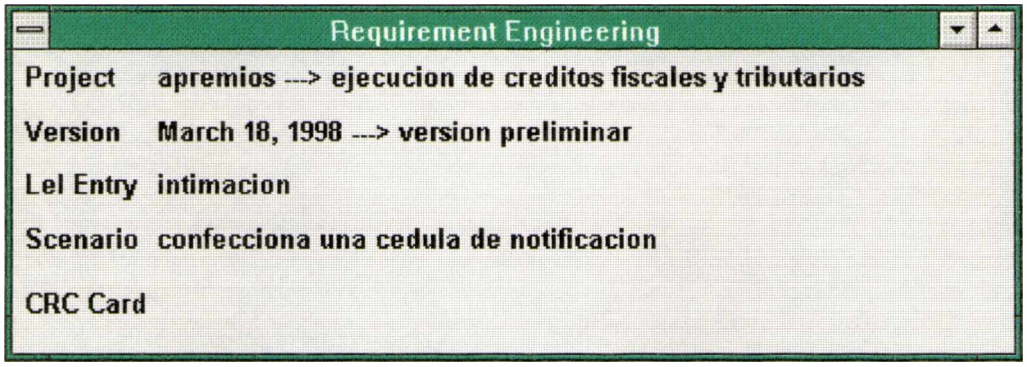
Las cuatro primeras opciones: full browser, half browser, summary y launcher, permiten elegir el tipo de ventana desde la cual se edita y navega la información. El full browser es la ventana de a continuación



El half browser es:



El summary browser tiene la siguiente apariencia:



La opción launcher, cierra la ventana de la vista, dejando solo el launcher. Otra opción del menú view es example, el cual crea un proyecto de ejemplo. Por último, con la opción exit, se cierra la aplicación. Todas las opciones del menú view se encuentran como botones del launcher, con la salvedad de que las opciones deshabilitadas no se muestran.

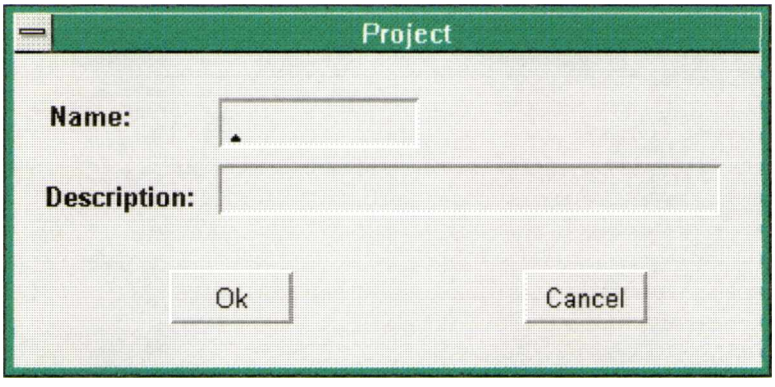
El resto de la funcionalidad de la herramienta se muestra a través de la vista full browser.

En ella se pueden ver cinco list boxes. Los mismos son Projects, Versions, Scenarios, LEL entries y CRC Cards. Cada list box tiene asociado un menú popup, que contiene las mismas opciones de la barra del menú. Así que el resto de la barra de menú se analiza desde el full browser.

2.2.2.Projects

El menú contextual asociado al list box de proyectos, cuando no hay ningún proyecto seleccionado, es el siguiente:

Al elegir la opción new se abre una ventana de diálogo para ingresar el nombre del proyecto y una descripción. El proyecto debe tener algún nombre y este no debe estar repetido. En cambio, la descripción es opcional.



Con la opcion load se puede cargar un proyecto desde disco agregándose a la lista de trabajo. Esta opción despliega una ventana de dialogo como la siguiente:

Junto al nombre de archivo se propone la extensón .ret y se puede agregar el path de localización del archivo, de hacer falta.

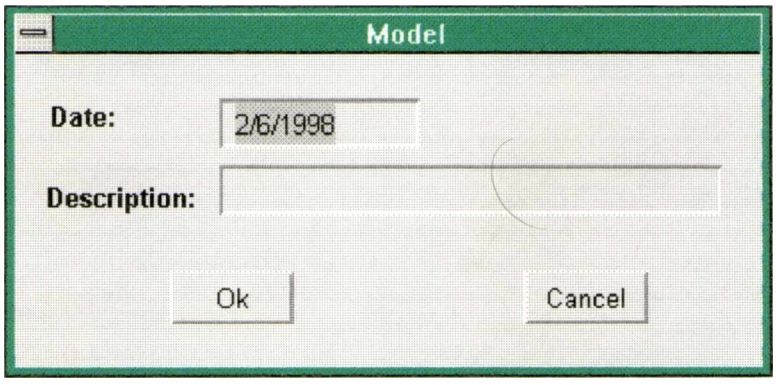
Cuando algún proyecto es seleccionado, el menú contextual que se despliega es, en cambio:

A la opción new y load ya mencionadas se agregan del, edit y save. Con del se borra el proyecto con todas sus versiones. Edit permite modificar el nombre y la descripción. Hay que tener encuentra que si se modifica el nombre no sea igual a otro que existe. Por último, save permite guardar un proyecto en un archivo en disco.

2.2.3. Versions

Para las versiones, si no hay ninguna creada el menú que se despliega es:

Con esta opción se abre una ventana de dialogo como la siguiente:



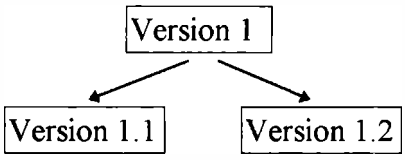
La fecha es obligatoria y la descripción es opcional. Por defecto se propone la fecha actual. Una vez ingresados los datos se crea la primera version.

Con la última version seleccionada el menú que se despliega es el siguiente:

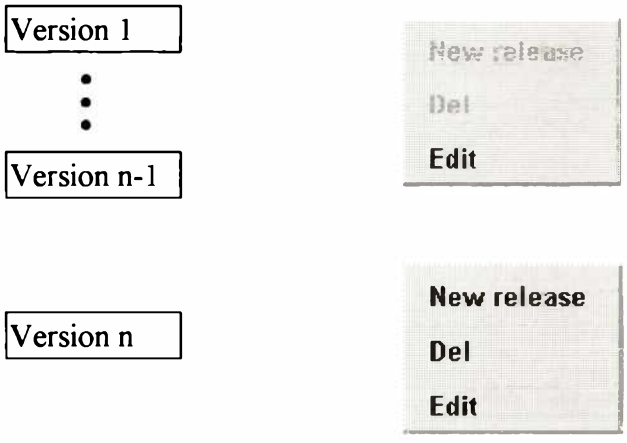
Con del se borra la version y todos sus elementos: LEL entries, Scenarios y CRC Cards. Con edit se puede modificar la fecha y la descripción. Y con export to html se generan páginas html con la información de la version.

Si la version seleccionada no es la última, el menú en cambio se muestra de la siguiente forma:

Las primeras dos opciones están deshabilitadas, impidiendo que se realice una nueva version a partir de la seleccionada, o se la borre. Las versiones deben seguir una evolución lineal. A partir de la primera version con new release se copia toda la información en una nueva. De esta forma se edita la nueva con los cambios y modificaciones necesarias. Ahora una nueva version se puede realizar sólo a partir de la última, no es posible obtener una nueva version de la primera. Una situación como la siguiente es indeseable y la aplicación impide su creación.



Lo que se desea es que sea lineal, por lo cual el menú asociado a cada version es el siguiente:



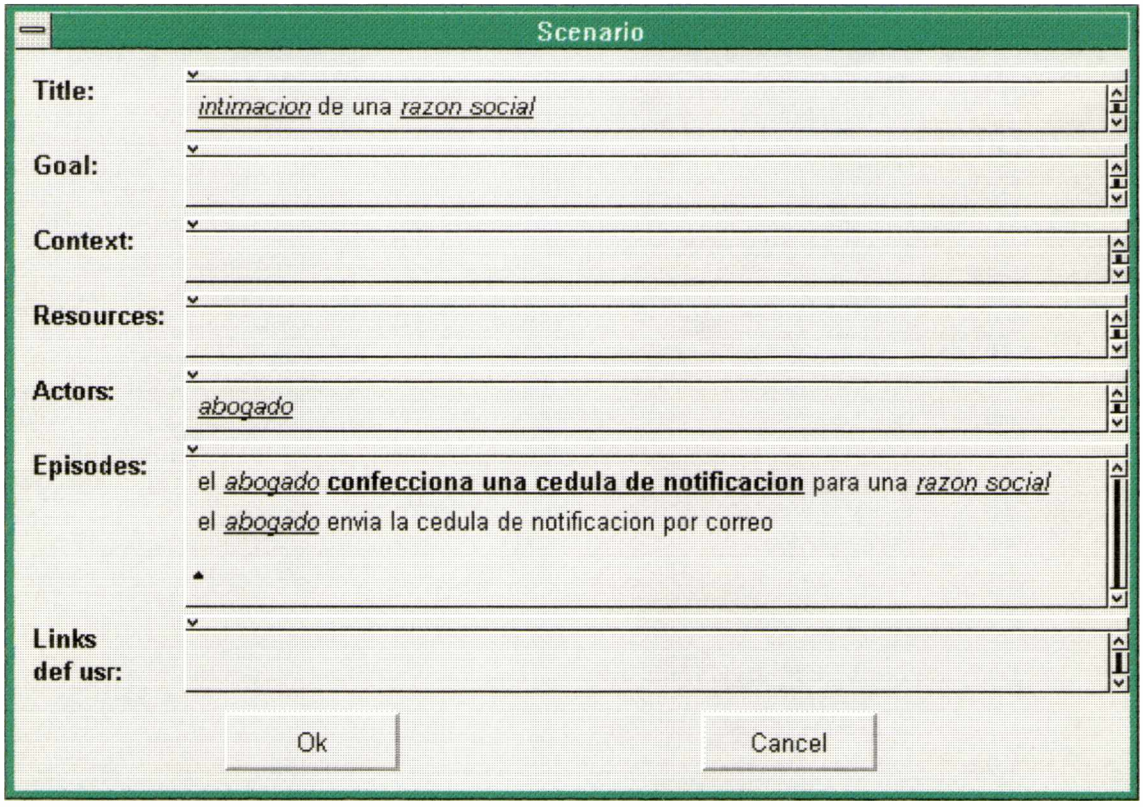
Además de permitir borrar o crear una nueva version solo a partir de la última, solo permite editar los elementos (LEL entries y scenarios, las CRC cards se derivan en forma automática y no se editan) de la última version. En todas las versiones anteriores a la última no se permite editar nada, ya que esa version tal y como se encuentra da origen a la siguiente y cualquier modificación no se reflejaría en las siguientes, quebrando la idea de evolución del modelo.

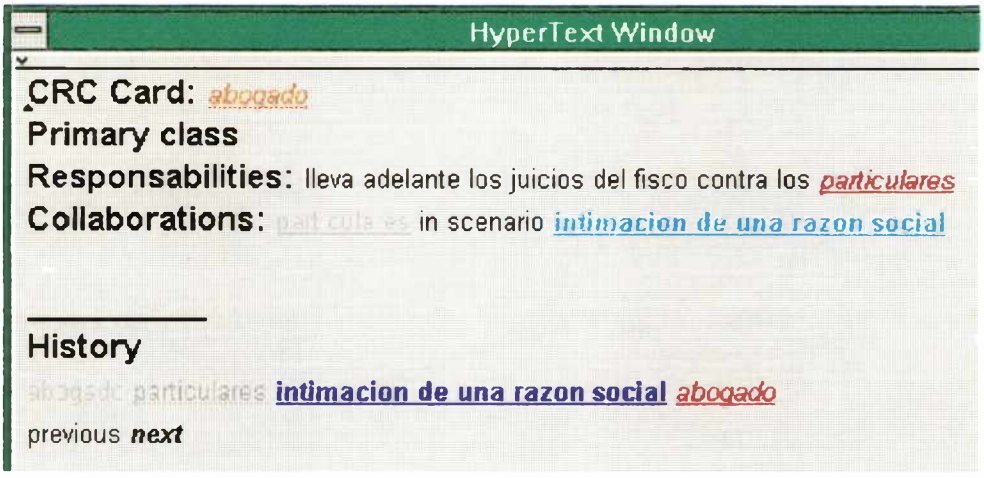
Y aunque se trate de la última version, si se han derivado las CRC cards, tampoco se permite la edición de LEL entries y scenarios. Como las CRC cards se derivan de la información de los scenarios y LEL entries, si se altera algo en ellos, se podría perder la consistencia, teniéndose CRC cards que no se justifican o apareciendo nuevas.

2.2.4.Links

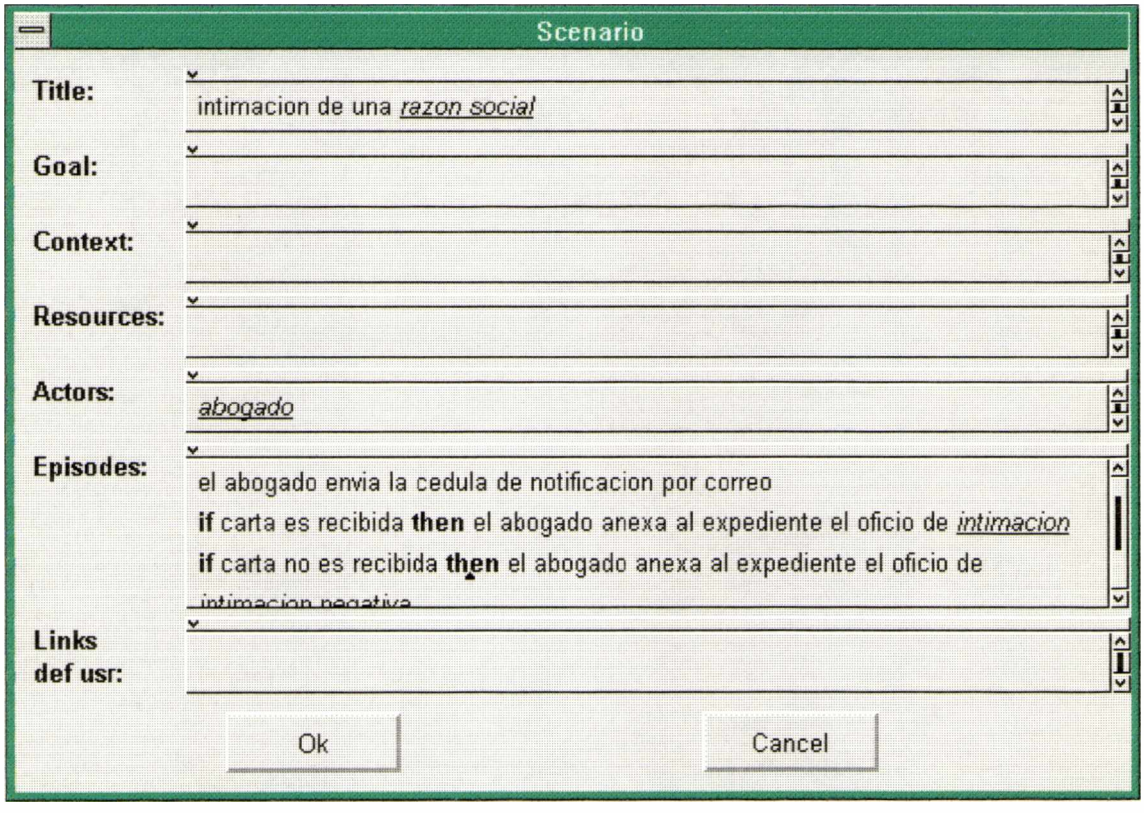
Los escenarios, los LEL entries y las CRC cards se relacionan por links. Estos poseen distinta apariencia en función de si se está navegando o editando, de si se está linkeando a un escenario, LEL entry o CRC card y si se está navegando si se visitó o no el link.

Durante la edición sólo se pueden definir links a escenarios y LEL entries, los links a CRC cards se derivan en forma automática. Los links se diferencian del texto normal (tanto en edición como navegación) en primer lugar por estar subrayados. Luego los links a LEL entries son en cursiva y los de escenarios están en negrita.





Cuando se navega, los links a escenarios y LEL entries siguen siendo subrayados y, con negrita y cursiva respectivamente. Se agregan los links a CRC cards que están simplemente subrayados. Y se les da color: los LEL entries son rojos, los scenarios azules y los CRC cards grises. En el caso que el link se haya visitado, el color es una tonalidad más clara.



Además, de los links los scenarios o LEL entries pueden contener tokens. Estos son texto con una sintaxis específica que no se puede editar. Por ejemplo, la descripción de los episodios de un escenario puede incluir frases como "if ... then ..." en donde if y then son tokens, se ingresan por medio del menú y no se pueden editar. Para los LEL entries los sinónimos se separan por "/", este también es un token.

Los tokens se identifican por estar en negrita (sin subrayado) y no poseer ningún color característico durante la navegación.

Tanto los links como los tokens no se pueden editar. Cuando el cursor está dentro de uno o hay marcado una porción de texto que lo incluya, la aplicación no acepta ninguna tecla salvo las de movimiento de cursor (flechas izquierda, derecha, arriba, abajo, home y end). Tampoco se puede editar el texto con las opciones del menú contextual, el cual deshabilita las opciones copy, cut y paste.

Para los links solo hay una operación que se puede realizar: pasar a normal. Para ello se debe marcar completamente el link, habilitándose la opción del menú. Hay que tener en cuenta que para crear los links se debe marcar completamente la palabra la aplicación no permite que un link empiece en la mitad de una palabra.

Para los tokens, las opciones de links no están disponibles. No es posible linkearlo ni pasarlo a normal.

2.2.5.LEL entries

El menú contextual para los LEL entries varia de la misma forma que el de los escenarios.

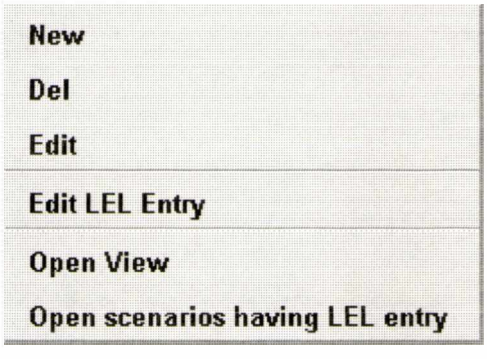
Si no se puede editar la información y no hay ninguno seleccionado no se muestra ningún menú.

Si no se puede editar la información pero hay uno seleccionado se muestra:

Open View (o doble click) muestra la información del LEL entry. Open scenario having LEL entry, por su parte, busca todos los escenarios que contienen al LEL seleccionado y permite navegarlos utilizando el recurso del contexto navegacional.

Si se puede editar la información y no hay ninguno seleccionado el menú que aparece es:

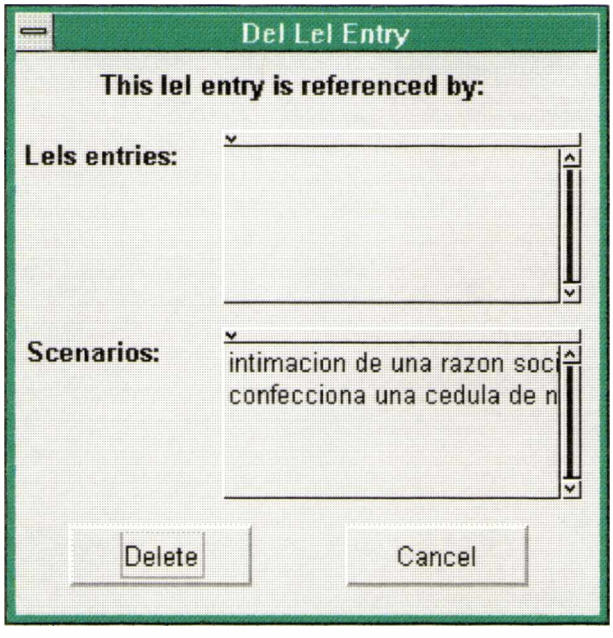
Por último, si se puede editar la información y hay algún LEL entry seleccionado el menú es:



Con new se crea un nuevo LEL entry. La información obligatoria que se debe ingresar es algún sinónimo y la nocion. El sistema no admite que dos entradas compartan sinónimos. El resto de la información a ingresar son los impactos. Para su edición cada menú contextual dispone de las opciones copy, cut y paste. Además incluyen opciones para hacer links hacia LEL entries (solo las nociones e impactos).

En el caso de los sinónimos, su menú posee las opciones de agregar sinónimo y borrar sinónimo. Con agregar sinónimo se agrega una barra "/" al comienzo, para que sirva de separador entre las palabras. Con borrar sinónimo, se elimina la palabra entre barras que se encuentra donde se encuentra el cursor.

Con Del se borra el LEL entry como así todas las referencias de otros LELs y escenarios hacia este. Sin embargo, antes de borrarlo, muestra una ventana de diálogo con las referencias.



Edit permite modificar la información de una entrada. Como es posible que se modifique algún sinónimo, si esto sucede, y el LEL entry es referenciado por otro, también se actualiza (bajo confirmación del usuario) el texto del link del escenario o LEL entry que referencia al modificado. Si el anchor era un sinónimo que el LEL entry no lo tiene más, se toma como anchor el primer sinónimo del LEL entry.

Con la opción Edit al igual que con New, cuando se acepta la entrada la aplicación realiza los links que estén definidos y no se hayan marcados a mano.

El menú también incluye las opciones edit LEL entries, open view y open view having LEL entry.

2.2.6.Scenarios

El menú contextual para los escenarios varía en función de si se puede editar la información y de si hay alguno seleccionado. Si no se puede editar la información (ya sea porque el escenario no es de la última version o hay CRC cards) y no hay ninguno seleccionado no se muestra ningún menú.

Si no se puede editar la información pero hay uno seleccionado se muestra una opción para navegar:

Con esta opción (o haciendo doble click sobre el escenario) se abre una ventana con la información del escenario para navegarla.

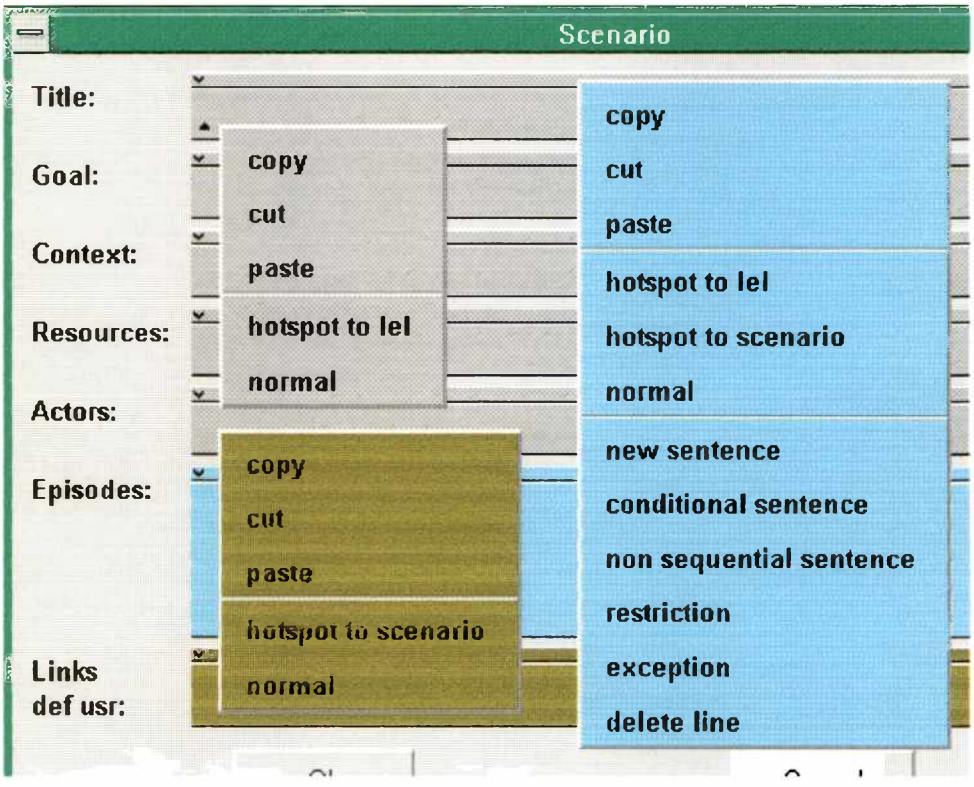
Si se puede editar la información y no hay ninguno seleccionado el menú que aparece es:

Por último, si se puede editar la información y hay algún escenario seleccionado el menú es:

Con new se crea un nuevo escenario. La única información obligatoria que se debe ingresar es el titulo, el cual debe ser distinto a todos los demás, la aplicación no admite dos títulos iguales. El resto de la información a ingresar es goal, context, resources, actors, episodes y links def usr. Para su edición cada menú contextual dispone de las opciones copy, cut y paste. Además incluyen opciones para hacer links hacia LEL entries y escenarios. Para title, goal, context, resources y actors sólo se permite hacer links hacia LEL entries. Y si una palabra de alguno de estos atributos se define como link, la cual está repetida en los demás, también se marca como link. Si la misma palabra estuviera en episodes o links def usr, no se copiaría. Para el atributo episodes los links

que se permiten hacer son hacia LEL entries y también hacia otros escenarios. Para los links def usr, solo se permite hacer links hacia escenarios.

Para los episodios, como su descripción implica utilizar oraciones con cierto formato, desde el menú se pueden elegir las mismas. Y como incluyen token los cuales no se pueden borrar, para eliminar alguna linea de los episodios, el menú también incluye una opción de borrar líneas.



Con Del se borra el escenario como así todas las referencias de otros escenarios hacia éste. Sin embargo, antes de borrar el escenario, muestra una ventana de diálogo con las referencias al escenario a borrar.

Edit permite modificar la información de un escenario. Como es posible que se modifique el titulo, si esto sucede, y el escenario es subescenario de otro, también se actualiza (bajo confirmación del usuario) el texto del link del escenario que referencia al modificado.

Antes de la edición

title: intimacion episodes: **confeccionar carta de noficacion**

Se edita el titulo del escenario

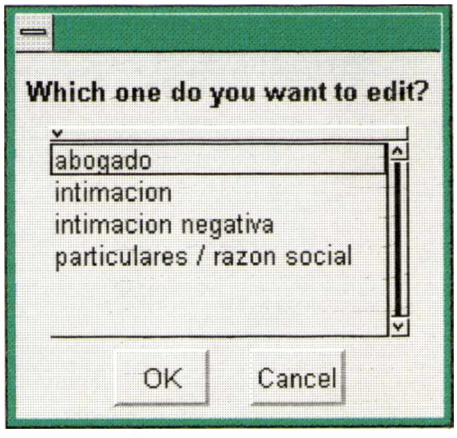
title: confeccionar cedula de noficacion episodes:

Se modifica el anchor

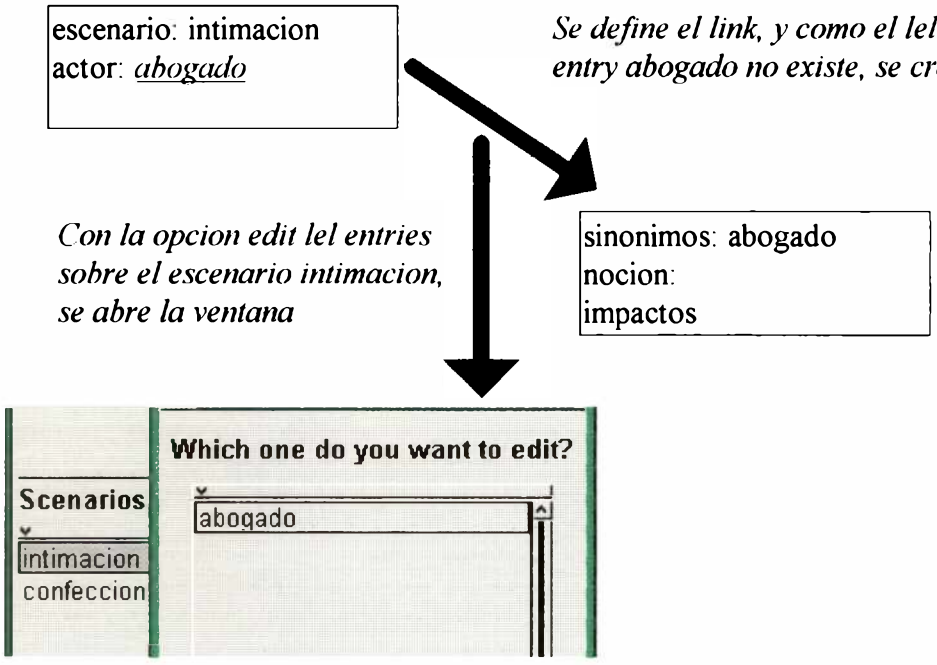
title: intimacion episodes: **confeccionar cedula de noficacion**

Con la opción Edit al igual que con New, cuando se acepta el escenario la aplicación realiza los links para las palabras y subescenarios que estén definidos y no se hayan marcados a mano.

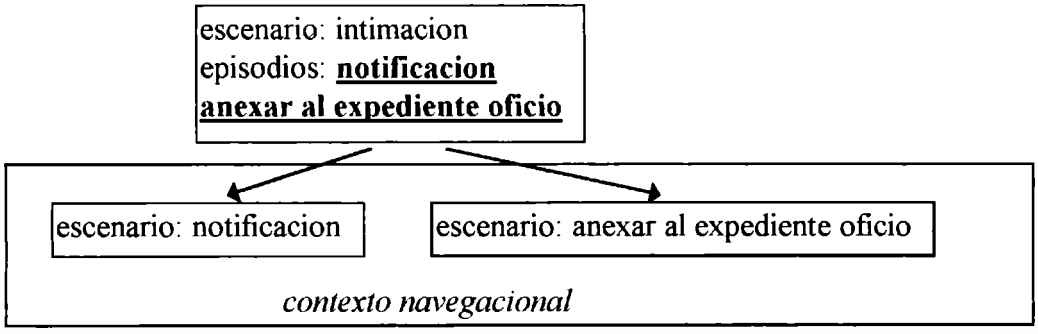
El menú de escenario también incluye las opciones edit subscenarios, edit LEL entries y edit links def usr. Cada una de estas opciones permiten editar las referencias del escenario seleccionado. Si se elige LEL entries, por ejemplo, se abre una ventana de diálogo con todos los LEL entries que son links del escenario, eligiendo uno, se lo puede editar.



Esta opción es útil para cuando se edita un escenario y se definen links hacia elementos que no están creados. Como los elementos se crean sin información, sólo con el titulo o un sinónimo, eligiendo esta opción, se pueden definir todos los elementos creados.



La última opción es open view. Con ella se abre una ventana para navegar la información del modelo. En la pantalla de navegación hay tres zonas. En la parte superior, se detalla la información del elemento seleccionado. Para este ejemplo, esta primera parte esta representada por el renglón scenario y el nombre. En la herramienta, se despliega toda la información. Al elegirse los links de esta sección se muestra la información de este nuevo link y se lo agrega al histórico. La historia de la navegación representa la segunda parte de la pantalla. En ella el elemento que está siendo visitado se muestra como link visitado (mas claro). También hay dos links para ir al anterior y siguiente. La tercera porción lo forma el contexto navegacional (ver Ilustración 24 en página 89). Si se elige un link a subescenario o a links def usr, se muestra esta tercera sección. La utilidad es recorrer todos los subscenarios de una forma rápida. Cuando se elige algún subscenario, se muestra la información de este y en la parte de navigational context se muestran todos los subscenarios que tenía el primero. De esta forma, se pueden visitar todos los subscenarios con las opciones previous y next. También es posible elegirlos directamente de la lista.



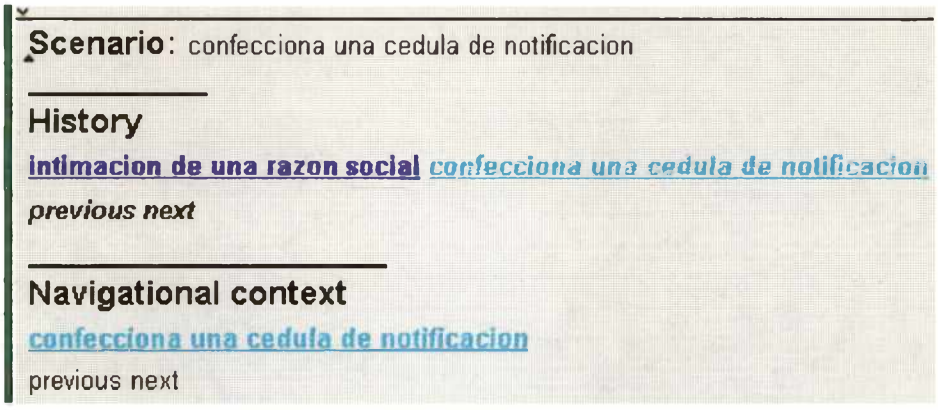


Ilustración 24. Areas en la pantalla de navegación

2.2.7. CRC cards

Las CRC cards no se pueden editar. La aplicación las genera en forma automática. Y para mantener la consistencia de la información, una vez derivadas las CRC cards no se permiten editar los escenarios o LEL entries.

El menú contextual varía en función de si se trata de la última version, de si hay CRC derivadas o no y de si hay o no seleccionada una tarjeta.

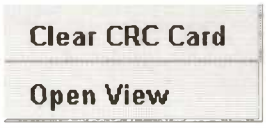
Si no es la última version y no hay ninguno seleccionado no se muestra ningún menú.

Si no es la última version, pero hay uno seleccionado se muestra:

Si es la última version y no hay tarjetas:

Con esta opción, valiéndose de los LELs y escenarios, y siguiendo el algoritmo que se describe en 1.4 se derivan las tarjetas CRC.

El menú correspondiente cuando se trata de la última version y hay tarjetas es:



Con clear CRC cards, se eliminan todas las tarjetas y se posibilita la edición de los escenarios y entradas del LEL.

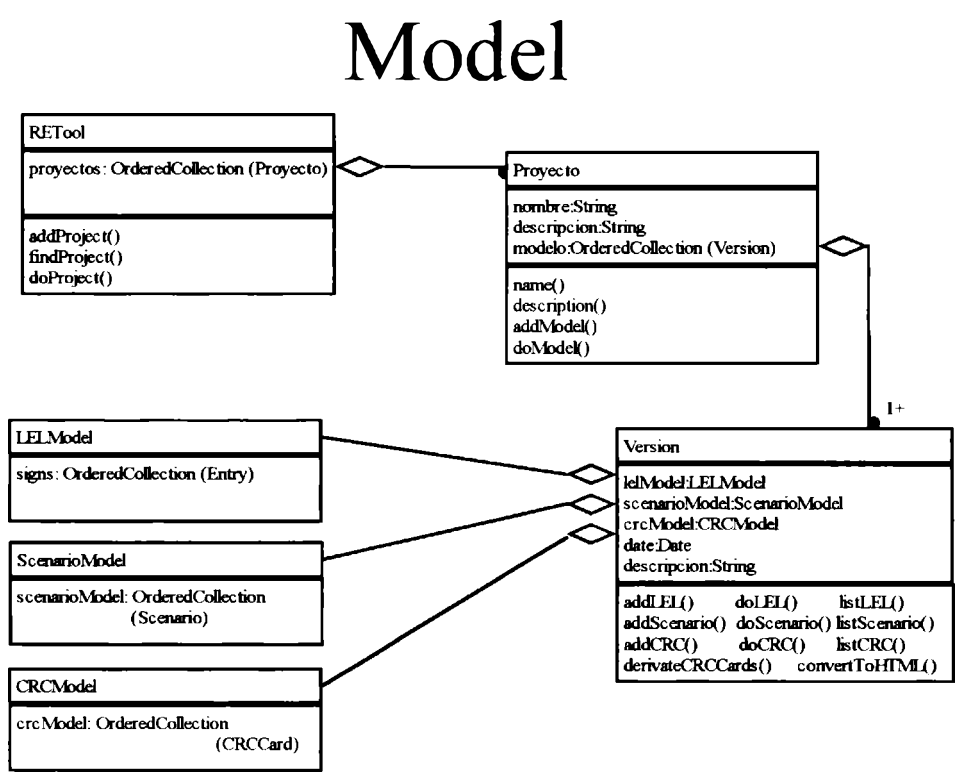
2.3. Diseño

Un buen diseño dota al soft de legibilidad y modificabilidad, haciendo factible el reuso y extensión.

2.3.1.Modelo de objetos de la herramienta

Los siguientes diagramas ilustran el modelo de la herramienta:

2.3.1.1.Diagrama principal



RETool (Requirements Engineering Tool): Repositorio de todos los proyectos que administra la herramienta.

Responsabilidades: Permite acceder y mantener el conjunto de elementos. Como la forma de identificar a los proyectos es por medio de su nombre, no permite que haya nombres repetidos.

Colaboraciones: Recibe pedidos de la interfase para acceder a la lista de proyectos.

Proyecto: Es una instancia del baseline. Cada proyecto identificado con un nombre, está formado por las distintas versiones de LELs, scenarios y CRCs.

Responsabilidades: Permite acceder y administrar el conjunto de versiones.

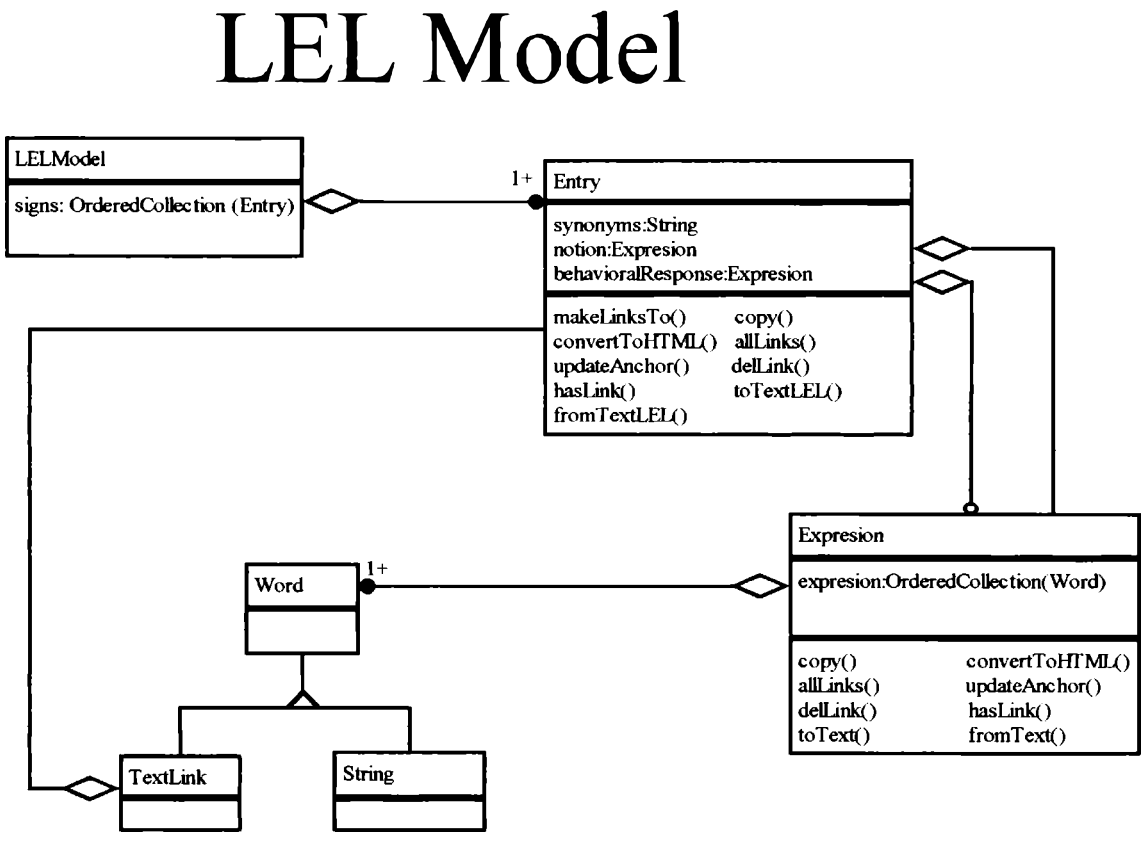
Colaboraciones: Recibe pedidos de la interfase para acceder a las distintas versiones.

Version: Representa cada paso en la evolución del baseline. A diferencia del modelo teórico planteado, en donde los cambios de configuración del LEL o scenarios se pueden producir arbitrariamente en el tiempo, la herramienta plantea eras (versiones) donde el cambio de era (version) permite cambios de configuraciones.

Responsabilidades: Administra el conjunto de LELs, scenarios y CRCs que la conforman. Conoce la estrategia para derivar las CRCs a partir de LELs y scenarios. Sabe como traducir la informacion a HTML.

Colaboraciones: Obtiene para la interfase elementos específicos dentro de la version.

2.3.1.2.Modelo del LEL



Entry: El Lexico Extendido del Lenguaje es un conjunto de frases o palabras, cada una es una instancia de Entry. Cada uno tiene sinónimos los que no se pueden repetir entre distintas entidades.

Responsabilidades: Crear links contra otras instancias, desahacer links, traducirse a HTML y texto.

Colaboraciones: Responde a la version sobre sus características y pide servicios a las expresiones.

Expresion: Conforma cada uno de los atributos de los LELs, scenarios y CRCs. La expresion contiene los links hacia otros elementos del modelo.

Reponsabilidades: Es quien en último término crea los links contra otras instancias, los deshace, se traduce a HTML y texto.

Colaboraciones: Resuelve los pedidos de los Entry y scenarios para manipular los links. Pide servicios a los TextLink para crearlos o modificar el anchor.

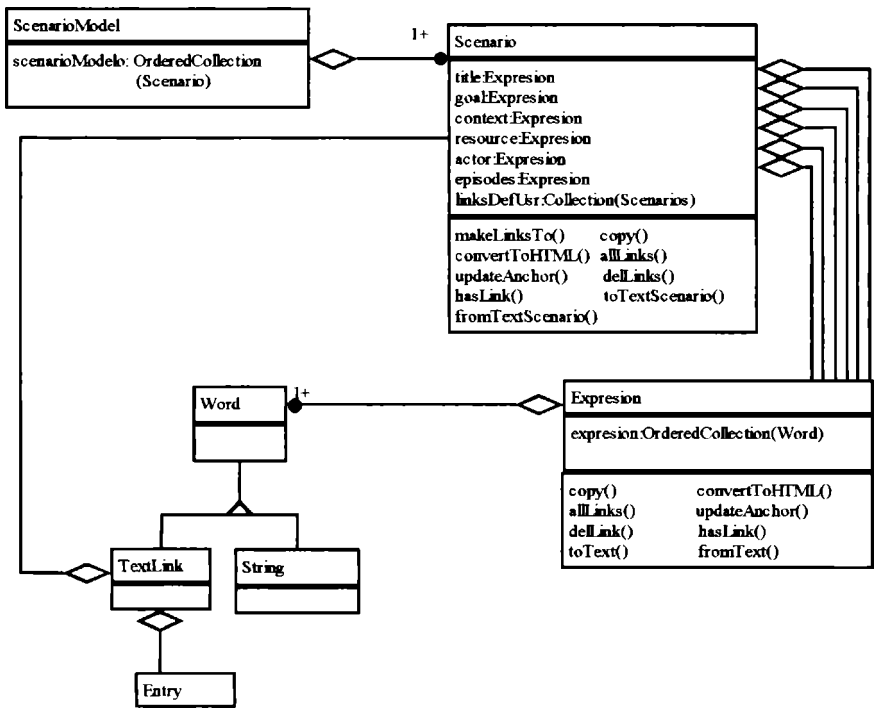
TextLink: Es el lazo de conexión entre las entidades del modelo. Cabe señalar que se preven ImageLink para futuras ampliaciones de la herramienta.

Responsabilidades: Sabe crearse con un anchor y conectándose a otra entidad. Responde acerca de sus características y permite modificar anchor o link.

Colaboraciones: Responde a las expresiones creando links o modificando sus atributos.

2.3.1.3.Modelo de escenarios

Scenario Model



Scenario: Describe una situación y el comportamiento a realizar. Se identifican quienes realizan las acciones y con que herramientas cuenta. Se lo identifica por un titulo y el mismo es único.

Responsabilidades: Crear links contra otras instancias, desahacer links, traducirse a HTML y texto.

Colaboraciones: Responde a la version sobre sus caracteristicas y pide servicios a las expresiones.

Expresion: Conforma cada uno de los atributos de los LELs, scenarios y CRCs. La expresion contiene los links hacia otros elementos del modelo.

Reponsabilidades: Es quien en último término crea los links contra otras instancias, los deshace, se traduce a HTML y texto.

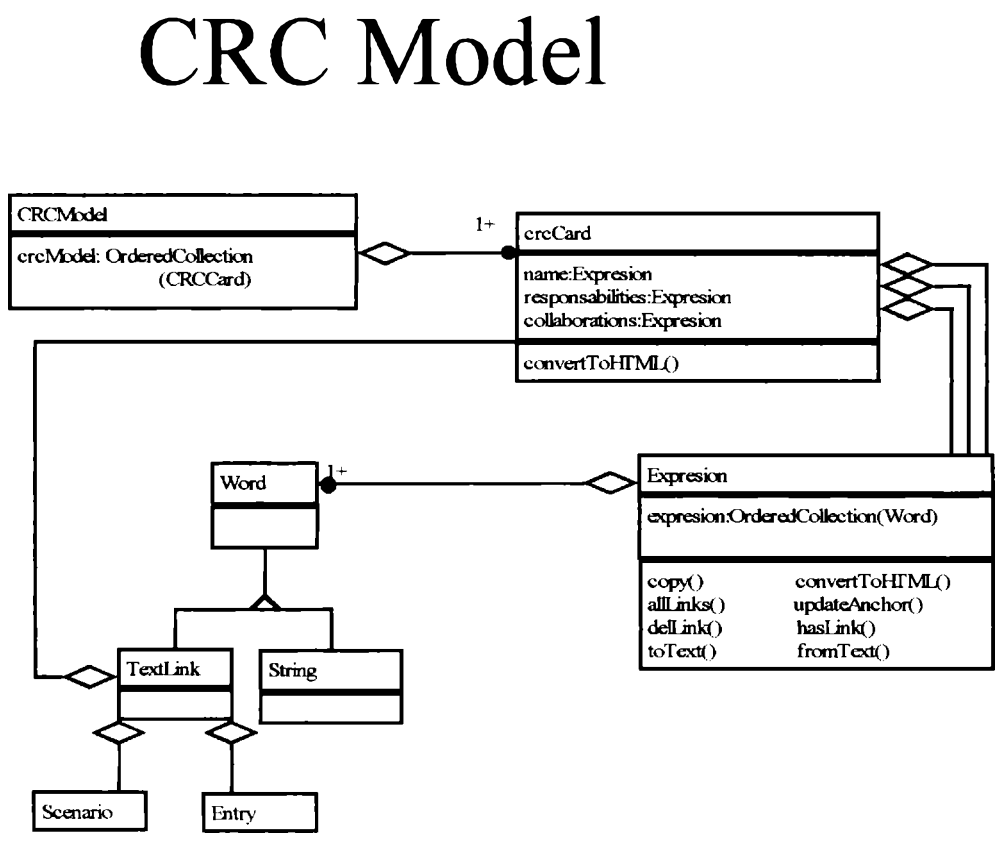
Colaboraciones: Resuelve los pedidos de los Entry y scenarios para manipular los links. Pide servicios a los TextLink para crearlos o modificar el anchor.

TextLink: Es el lazo de conexión entre las entidades del modelo. Cabe señalar que se preven ImageLink para futuras ampliaciones de la herramienta.

Responsabilidades: Sabe crearse con un anchor y conectándose a otra entidad. Responde acerca de sus características y permite modificar anchor o link.

Colaboraciones: Responde a las expresiones creando links o modificando sus atributos.

2.3.1.4.Modelo de tarjetas CRC



CRC: Contiene la información de las clases que se deben crear para implementar el sistema cuyos requerimientos acumulan LEL y scenarios. Las CRCs se identifican por un nombre de clase. Contienen las responsabilidades (metodos a implementar) y colaboraciones (clases con las cuales se comunica).

Responsabilidades: Sabe crearse con links contra otras instancias y traducirse a HTML.

Colaboraciones: Responde a la version que le solicita su creación de cierta forma.

Expresion: Conforma cada uno de los atributos de los LELs, scenarios y CRCs. La expresion contiene los links hacia otros elementos del modelo.

Reponsabilidades: Es quien en último término crea los links contra otras instancias, los deshace, se traduce a HTML y texto.

Colaboraciones: Resuelve los pedidos de los Entry y scenarios para manipular los links. Pide servicios a los TextLink para crearlos o modificar el anchor.

TextLink: Es el lazo de conexión entre las entidades del modelo. Cabe señalar que se preven ImageLink para futuras ampliaciones de la herramienta.

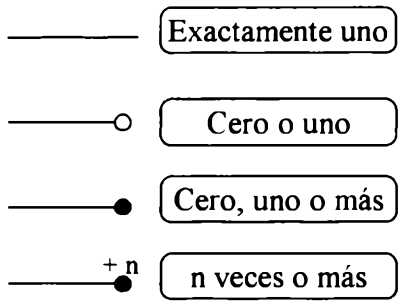
Responsabilidades: Sabe crearse con un anchor y conectándose a otra entidad. Responde acerca de sus características y permite modificar anchor o link.

Colaboraciones: Responde a las expresiones creando links o modificando sus atributos.

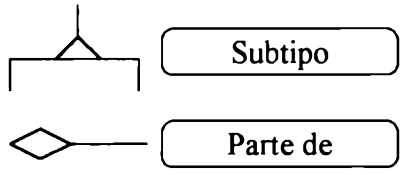
2.3.1.5.Comentario del modelo

2.3.1.5.1.Símbolos utilizados en la descripción del modelo

Cardinabilidad



Relacion estructural



2.3.1.5.2.Links

Con respecto a los links que están presentes en los modelos del LEL, Scenarios y CRC, se ha adoptado la siguiente postura. El texto que describe las características de una entrada del LEL, de un escenario o de una tarjeta CRC está compuesto por String y TextLinks (son anchors que contienen a la referencia en sí: una entrada del LEL, escenario o tarjeta CRC).

En el LEL Model, las nociones e impactos de una entrada deben contener referencias a otras entradas (principio de circularidad), así que las entradas tienen links hacia otras entradas.

En el Scenario Model, se pueden presentar tres tipos de links distintos.

- En primer lugar, en la descripción de los atributos de un escenario (titulo, objetivo, contexto, recursos, actores y episodios) se deben utilizar términos definidos en el LEL, por lo que, allí habrá links hacia el LEL.
- En segundo lugar, los episodios de un escenario, pueden ser a su vez, subescenarios, así que en los episodios se pueden tener links a otros escenarios.
- En tercer y último lugar, se encuentran los links definidos por el usuario, de modo que cada escenario estará conectado con otros (estos otros definidos en forma arbitraria)

Por su parte, en el CRC Model, los links que pueden aparecer son:

- Tanto el nombre como las responsabilidades, tienen referencias al LEL. El nombre es un actor que tiene que tener definición en el LEL (revisar método para derivar CRC) y las responsabilidades son los impactos del LEL que por pertenecer al LEL tienen referencias.
- Los servicios son links hacia otras CRC, ya que indican con que otras clases colabora.

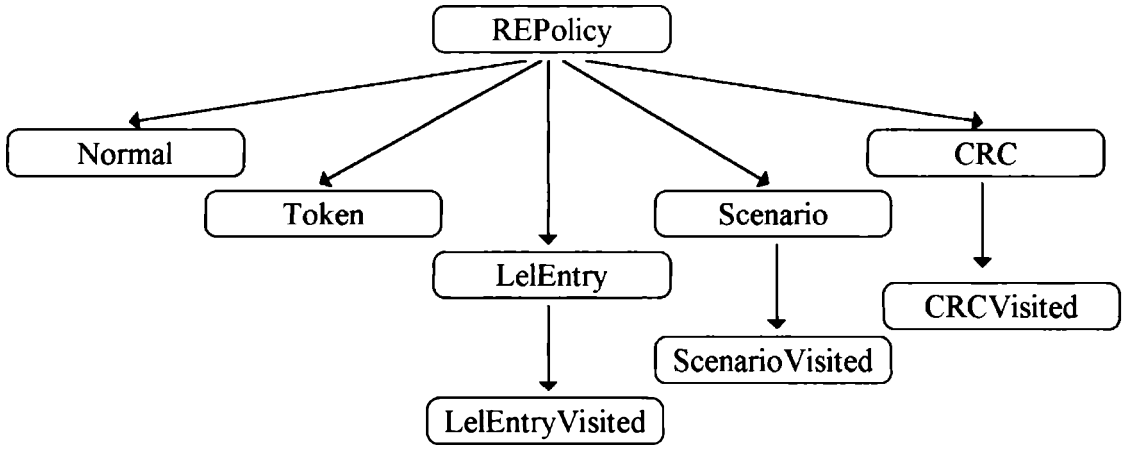
- Y el contexto son referencias hacia los escenarios, puesto que el contexto identifica a los escenarios en los que aparecen las clases con las que colabora determinadas por los links del punto anterior.

2.3.1.6.Estructura de REPolicy

Esta estructura surge de la necesidad de satisfacer la premisa de que la herramienta sea modificable.

VisualWorks incorpora nativamente una jerarquía Policy, la cual se encarga de gestionar la apariencia que tiene la interfase. Como permite configurar que la apariencia sea macintosh, windows, y algunas más, se necesita de un modo de dibujar las ventanas con el estilo adecuado. Cuando VisualWorks necesita dibujar algún elemento le pide al Policy por las características del elemento, que difieren en los distintos estilos de interfase. Siguiente la misma política, en la aplicación se necesitan mostrar distintos tipos de links, así que se dispone de una jerarquía REPolicy para tal fin.

La clase REPolicy y sus subclases son las encargadas de definir los atributos del texto a mostrar durante la edición y la navegación. Esta clase conoce de que forma se deben mostrar cada tipo de links: los visitados y no visitados, los tokens y el texto normal. La jerarquía es la siguiente:



De esta forma, teniendo los atributos de los links centralizados en la jerarquía REPolicy, cualquier modificación que se quiera hacer de estilo de letra o color se logra accediendo solamente a la clase correspondiente.

2.3.2.Decisiones de diseño

El único elemento que merece ser mencionado es la forma en que se almacenan los links. Los mismos se guardan como atributos propios del texto que es editado.

VisualWorks posee la clase Text. Un objeto Text esta compuesto por string el cual posee ciertos atributos. Dos variables de instancias contiene esta clase, una es string y la otra es runs (esta ultima almacena los atributos). Los componentes visuales que soportan la edición de texto, necesitan de una instancia de Text para almacenar la información. Así es que durante la edición de un scenario o LEL entry, si se define que una palabra es el anchor de un link, el objeto que se relaciona con esa palabra se almacena como atributo mismo de la palabra. Es decir, la palabra al convertirse en link va a tener atributos distintos, va a estar subrayada (por ejemplo) y otro atributo que va a contener es el objeto al que se linkea.

Entonces, los links se almacenan como parte del texto mientras se edita la información. Pero, cuando el escenario es almacenado, cada objeto de la clase texto se transforma en una expresion. Esta expresion esta formada por una secuencia de Strings y TextLinks (como puede verse en Modelo de objetos de la herramienta en pagina 67). El texto que es llano se transforma como string. Y para el que es un link, se instancia un TextLink, el cual contienen información del string y el objeto que es el link de este.

2.3.3.Clase REInterfase

REInterface es la clase que define la interfase de RETool. Por medio de los siguientes escenarios se detalla la funcionalidad que tiene esta clase

Nombre: nuevo proyecto
Recursos: lista de proyectos, proyecto seleccionado
Episodios:
1.[[abrir formulario de proyecto]]. Excepcion: si se cancela, se termina el escenario
2.Crear un proyecto con los datos ingresados.
3.Definirle un modelo nuevo
4.Almacenar el proyecto en RETool
5.Tomar como proyecto seleccionada el ingresado
6.[[mostrar lista de modelos]]

Nombre: abrir formulario de proyecto
Recursos: lista de proyectos, proyecto seleccionado
Episodios:
1.Abrir ventana de dialogo
2.Editar nombre y descripción
3.Aceptar o cancelar. Excepcion: Si el nombre del proyecto está repetido, no permitir aceptar.
4.Cerrar ventana de dialogo

Nombre: borrar proyecto
Recursos: lista de proyectos, proyecto seleccionado
Episodios:
1.Confirmar decisión. Excepcion: Si no se confirma la decisión, se cancela la operación
2.Borrar el proyecto de RETool
3.[[Desseleccionar Proyecto]]
Restricción: proyecto seleccionado debe tener un valor

Nombre: modificar proyecto
Recursos: proyecto seleccionado, lista de proyectos
Episodios:
1.Mostrar ventana de dialogo con los datos del proyecto seleccionado. Excepcion: Si se cancela la operación se termina el escenario.
2.Cambiar el nombre y descripción del proyecto seleccionado por los ingresados.
Restricción: proyecto seleccionado debe tener algún valor.
Restricción: el nombre nuevo no debe ser igual a otro ya ingresado.

Nombre: seleccionar proyecto
Recursos: lista de proyectos, proyecto seleccionado

Episodios:
1.El proyecto clickeado se toma como proyecto seleccionado
2.[[Mostrar listas de modelos]]

Nombre: desseleccionar proyecto
Recursos: proyecto seleccionado, lista de proyectos
Episodios
1.Inicializar proyecto seleccionado
2.[[Quitar contenido de lista de modelos]]
Restricción: proyecto seleccionado debe tener algún valor

Nombre: crear modelo: (en realidad lo que se crea es una version del modelo)
Recursos: proyecto seleccionado, lista de modelos, modelo seleccionado
Episodios:
1.[[abrir formulario de modelos]]. Excepcion: Si se cancela, se termina el escenario
2.Crear una Version nueva
3.Definirle fecha y descripción con las ingresadas
4.Definirle modelos de LEL, escenarios y CRC nulos.
5.Guardar esa version en el modelo de proyecto seleccionado
6.Asignar a modelo seleccionado el modelo ingresado
7.[[mostrar lista de escenarios]]
8.[[mostrar lista de LELs]]
9.[[mostrar lista de CRCs]]
Restricción: proyecto seleccionado debe tener algún valor.

Nombre: modelForm open
Recursos: modelList, modelSelected
Episodios:
1.Abrir ventana de dialogo
2.Editar fecha y descripción
3.Aceptar o cancelar. Excepcion: Si no se ingresa fecha, no permitir aceptar.
4.Cerrar ventana de dialogo

Nombre: Borrar Modelo:
Recursos: lista de modelos, modelo seleccionado, proyecto seleccionado
Episodios:
1.Confirmar la decisión. Excepción: Si no se confirma la decisión, se cancela la operación
2.Quitar el modelo seleccionado de la lista del proyecto seleccionado.
3.[[mostrar lista de modelos]]
Restricción: modelo seleccionado debe tener algún valor

Nombre: Modificar Modelo
Recursos: modelo seleccionado, proyecto seleccionado
Episodios:
1.[[abrir formulario de modelo]]. Excepción: se puede cancelar la operación abortando las modificaciones.
2.actualizar la información del modelo seleccionado con los datos ingresados.

Restricción: modelo seleccionado debe tener algún valor

Nombre: Seleccionar Modelo
Recursos: modelo seleccionado, lista de modelos
Episodios
1.Asignar a modelo seleccionado el modelo clickeado.
2.[[mostrar lista de LELs]]
3.[[mostrar lista de escenarios]]
4.[[mostrar lista de CRCs]]
Restricción: proyecto seleccionado debe ser distinto de nulo.

Nombre: Desseleccionar Modelo
Recursos: modelo seleccionado, lista de modelos
Episodios
1.modelo seleccionado se hace nulo
2.[[mostrar lista de escenarios]]
3.[[mostrar lista de LELs]]
4.[[mostrar lista de CRCs]]

Nombre: nueva version
Recursos: lista de modelos, modelo seleccionado, proyecto seleccionado
Episodios:
1.[[abrir formulario de modelo]]. Excepcion: Si se cancela, se termina el escenario.
2. Crear una Version nueva.
3. Definirle la fecha y descripcion ingresadas.
4. Crear lista de CRC nueva.
5. Para cada LELEntry del modelo seleccionado, [[copiar LELEntry]] y guardarla en el modelo nuevo.
6. Para cada scenario del modelo seleccionado, [[copiar scenario]] y guardarlo en el modelo nuevo.
7. Para cada LELEntry y scenario del modelo seleccionado: [[copiar links]]

Nombre mostrar lista de modelos
Recursos: proyecto seleccionado, lista de modelos, modelo seleccionado
Episodios
1.IF proyecto seleccionado es nulo THEN lista de modelos es una lista vacía
2.IF proyecto seleccionado no es nulo THEN lista de modelos es la lista de modelos del proyecto seleccionado
3.modelo seleccionado es nulo
4.[[mostrar lista de escenarios]]
5.[[mostrar lista de LELs]]
6.[[mostrar lista de CRCs]]

Nombre: Crear escenario
Recursos: lista de escenarios, escenario seleccionado, elementos creados
Episodios:
1.Crear un escenario nuevo.

2.[[abrir formulario de escenario]]. Excepción: se puede cancelar la edición, borrando el escenario creado (con sus links) y los LEL y subscenarios creados para estos links.
3.[[hacer links con LELs y escenarios]]
4.Agregar escenario a lista de escenarios.
5.escenario seleccionado es el escenario agregado
Restricción: modelo seleccionado debe ser distinto de nulo

Nombre: abrir formulario de escenario
Recursos: escenario, lista de LELs, lista de escenarios, elementos creados
Episodios:
1.# Editar: titulo, objetivo, contexto, recursos, actor y episodios.
2.[[Linkear palabra a LEL]]
3.[[Linkear palabra a escenarios]]#
4.Aceptar la edición del escenario. Excepcion: En cualquier momento se puede cancelar la edición. Restriccion: Si el nombre del escenario esta en lista de escenarios no permitir aceptar.

Nombre: Linkear palabra a escenario
Recursos: lista de escenario, elementos creados
Episodios:
1.Buscar un escenario con titulo igual a lo marcado, de lista de escenario y elementos creados.
2.IF existe THEN linkear palabra a ese escenario.
3.IF no existe THEN crear escenario nuevo. Agregarlo a elementos creados. Linkear palabra a ese escenario.

Nombre: linkear palabra a LEL
Recursos: lista de LELs, elementos creados
Episodios:
1.Buscar la entrada del LEL igual que contenga un sinónimo a lo marcado, de lista de LELs y elementos creados.
2.IF existe THEN Linkear palabra a esa LELEntry.
3.IF no existe ninguno THEN crear LELEntry nuevo. Agregarlo a elementos creados. Linkearlo a ese.

Nombre: deshacer link
Recursos: elementos creados
Episodios:
1.Tomar el scenario o LELEntry del link a deshacer
2.IF link esta en elementos creados THEN borrarlo de elementos creados.
3.Cambiar atributos del tipo de letra del link a normal.

Nombre: borrar escenario
Recursos: lista de escenarios, escenario seleccionado
Episodios:
1.Buscar los links definidos por el usuario que apunten a este subescenario.
2.IF hay algún link THEN informar las referencias a este escenario
3.Buscar los escenarios de los que es subescenario

4.IF hay algún link THEN informar las referencias a este escenario
5.Confirma si se quiere borrar el escenario
6.IF hay alguna referencia THEN borrar de los otros scenarios las referencias a este.
7.IF era subescenario THEN quitar el anchor de sus superescenarios
8.[[Definir contenido lista de escenarios]]
9.Restricción: escenario seleccionado no debe ser nulo.

Nombre: editar escenario
Recursos: escenario seleccionado
Episodios:
1.[[abrir formulario de escenario]]
2.IF se modifica el titulo y es subescenario de otro THEN cambiar el texto de los episodios con el nuevo titulo.
3.Actualizar la lista de escenarios.
4.Dejar al mismo escenario como seleccionado.
5.[[hacer link con LELs y escenarios]]
Restricción: escenario seleccionado no debe ser nulo. Verificar que el titulo del escenario no este repetido.

Nombre: editar subescenario
Recursos: escenario seleccionado
Episodios:
1.Tomar del escenario todos los subescenarios.
2.Mostrar la lista de subescenarios.
3.Permitir elegir un subescenario.
4.[[Editar escenario]]
5.Restricción: escenario seleccionado no debe ser nulo.

Nombre: editar LEL entry de un escenario
Recursos: escenario seleccionado
Episodios:
1.Tomar del escenario todos las entradas del LEL.
2.Mostrar la lista
3.Permitir elegir una
4.[[Editar entrada del LEL]]
5.Restricción: escenario seleccionado no debe ser nulo.

Nombre: abrir vista de escenario
Recursos: escenario seleccionado, contexto navegacional
Episodios:
1.Abrir una ventana de dialogo.
2.Mostrar la información hypertextual del escenarios.
3.IF se elige algún links THEN cerrar la ventana actual. Definir contexto navegacional [[mostrar escenario o LEL correspondiente]]
4.IF se elige siguiente o anterior THEN tomar el correspondiente del navigationalContext [[abrirlo]].
Restricción: escenario seleccionado no debe ser nulo.

Nombre: seleccionar escenario
Recursos: lista de escenarios, escenario seleccionado
Episodios:
1.escenario seleccionado es el escenario clickeado

Nombre: desseleccionar escenario
Recursos: lista de escenarios, escenario seleccionado
Episodios:
1.escenario seleccionado es nulo.
Restricción: escenario seleccionado debe ser distinto de nulo.

Nombre quitar contenido de lista de escenarios
Recursos: escenario seleccionado, lista de escenarios
Episodios:
1.vaciar lista de escenarios.
2.al escenario seleccionado asignarle nulo.

Nombre: definir contenido de lista de escenarios
Recursos: escenario seleccionado, lista de escenarios, modelo seleccionado
Episodios:
1.asignar a lista de escenarios los escenario del modelo seleccionado
2.al escenario seleccionado asignarle nulo.

Nombre: crear LEL, borrar LEL, modificar LEL, abrir LEL entry, crear LEL entry, copiar, seleccionar LEL entry, desseleccionar LEL entry, quitar contenido de lista de LELs, definir contenido de lista de LELs, linkear palabra a LEL
Ídem a los escenarios.

Nombre: seleccionar CRC, desseleccionar CRC, quitar contenido de lista de CRCs, definir contenido de lista de CRCs
Ídem scenarios.

Nombre: derivar CRC
Recursos: lista de escenarios, lista de LELs, lista de CRCs
Episodios:
1.Tomar todos los actores de los escenarios que son links hacia el LEL.
2.Para cada actor, tomar las responsabilidades de los impactos del LEL.
3.Para cada término de las responsabilidades de estas clases (primarias) que están en el LEL, definirla como clase secundaria.
4.Para cada clase secundaria tomar las responsabilidades de los impactos del LEL.
5.Para cada clase obtenida, se revisan todos los escenarios, buscando en los episodios con que otra clase definida colabora.
Restricción: Una vez derivadas las clases, se impide que se modifiquen los modelos de LEL y scenarios. Solo se puede consultar la información con abrir.

Nombre: abrir CRC
Recursos: CRC seleccionada
Episodios:

1.Mostrar la CRC card como hypertexto.
2.IF se elige algún anchor THEN cerrar ventana [[abrir link correspondiente]]

2.3.4.Clase Scenario, LEL Entry, Expresion

Nombre: copiar LEL entry
Recursos: LEL entry
Episodios:
1. Crear LEL entry nueva.
2. Para cada expresion que forma el LEL [[copiar la expresion]]

Nombre: copiar escenario
Recursos: escenario
Episodios:
1. Crear escenario nuevo.
2. Para cada expresion que forma el scenario [[copiar expresion]]

Nombre: hacer links con LELs y escenarios
Recursos: escenario, lista de LELs, lista de escenarios
Episodios:
1. Para cada elemento de la lista de escenarios:
1.1. Ver si el scenario nuevo puede ser subescenario.
1.2. IF puede THEN hacer el link.
2. Para cada episodio del scenario:
2.1. Buscar si hay subescenarios definidos.
2.2. IF hay THEN hacer el link.
3. Para cada palabra del scenario no link:
3.1. Verificar si tiene definicion en el LEL.
3.2. IF tiene THEN hacer el link.

Nombre: copiar expresion
Recursos: expresion
Episodios:
1. Devolver el texto que conforma la expresion, sin los links.

Nombre: copiar links
Recursos: escenario o LEL entry copiado, modelo copiado.
Episodios:
1. Para cada link del objeto.
2. Ubicar el objeto en la nueva version.
3. Hacer el link entre las copias.

3.Bibliografía

[Antón 94] Antón, Annie L. **Goal-Based requirements analysis.**

[Benner 93] Benner, Kevin M.; Feather, Martin S., Johnson W.; Zorman, Lorna A. **Utilizing scenarios in the software development process.** En proceedings of the 8th knowledge-based software engineering conference (KBSE 93), IEEE. Septiembre 1993.

[Booch 94] Booch Grady. **Scenarios.** Revista Road Volumen 1 numero 3 paginas 3 a 6. Septiembre - Octubre 1994

[Coad 91] Coad, Peter; Yourdon, Edward. **Object oriented design.** Yourdon Press, Prentice Hall Building 1991.

[Coad 95] Coad, Peter; North, David; Mayfield, Mark. **Object models strategies, patterns & applications.** Yourdon Press, Prentice Hall Building 1995.

[Firesmith 94] Firesmith, Donal G. **Modeling the dynamic behavior of system, mechanisms, and classes with scenarios.** Revista Road Volumen 1 numero 2 paginas 32 a 36. Julio - Agosto 1994.

[Ghezzi 91] Ghezzi, Carlo; Jazayeri, Mehdi; Mandrioli, Dino. **Fundamentals of Software Engineering.** Prentice-Hall International Editions. 1991.

[Hopkins 95] Hopkins, Trevor; Horan, Bernard. **Smalltalk: An introduction to application development using Visual Works.** Prentice Hall. 1995.

[Jacobson 94] Jacobson, Ivar. **Basic use-case modeling.** Revista Road volumen 1 numero 2 paginas 15 a 19. Julio-Agosto 1994

[Jacobson 94b] Jacobson, Ivar. **Basic use-case modeling.** Revista Road volumen 1 numero 3 paginas 7 a 9.. Septiembre-Octubre 1994

[Jacobson 94c] Jacobson, Ivar. **The Object Advantage. Addison Wesley Publishing Company.** 1994.

[Leite 90] Sampaio do Prado Leite, Julio Cesar; Moreira Franco, Ana Paula. **O uso de hipertexto na elicitação de linguagens da aplicação.** IV Simpósio brasileiro de engenharia de software, Águas de São Pedro, SP. Mayo 1990.

[Leite 93] Sampaio do Prado Leite, Julio Cesar. **Eliciting requirements using a natural lenguage. Based approach: The case of the meeting scheduller problem.** Cedido por el autor. Marzo 1993.

[Leite 94] Sampaio do Prado Leite, Julio Cesar; Moreira Franco, Ana Paula. **Uma estratégia de suporte à engenharia de requisitos.** XIX Seminario integrado de software e hardware.

[Leite 95]Sampaio do Prado Leite, Julio Cesar. **Recovering business rules from structured analysis specifications.** Second international working conference on reverse engineering. 1995.

[Leite 95b] Sampaio do Prado Leite, Julio Cesar; Albuquerque Oliviera, Antonio de Pádua. **A client oriented requirements baseline.** RE'95, páginas 108-115. IEEE Computer Society Press. 1995.

[Leite 97] Sampaio do Prado Leite, Julio Cesar; Rossi, Gustavo. **Enhancing a requirements baseline with scenarios.** RE'97, páginas 44-53. IEEE Third International Computer Society Press. 1997.

[Leite 97b] Sampaio do Prado Leite, Julio. **Use of scenarios in software development.** (Cedido por el autor)

[Leite 97c] Sampaio do Prado Leite, Julio; Breitman, Karin Koogan. **Scenario evolution: observation from a case study.** (Cedido por el autor)

[Leite 97d] Sampaio do Prado Leite, Julio; Leonardi, Carmen; Rossi, Gustavo. **Deriving object-oriented specification from external scenarios.** (Cedido por el autor)

[Leonardi 96] Leonardi, Carmen. **Estrategias basadas en escenarios para el desarrollo de software.** Informe de beca. Febrero 1996

[Potts 94] Potts, Colin; Takahash, Kanji; Anton, Annie. **Inquiry based requirements analysis**. IEEE Software, páginas 21-32. Marzo 1994.

[Potts 95] Potts, Colin. **Using schematic scenarios to understand user needs**. Diciembre 1995.

[Regnell 95] Regnell, Björn; Kimbler, Kristofer; Wesslén, Anders. **Improving the use case driven approach to requirements engineering.** Second IEEE International Symposium on Requeriments Engineering, páginas 40-47. Marzo 1995.

[Riel 96] Riel, Arthur J. **Object Oriented Design Heuristics.** Addison Weley Publishing company, inc. 1996.

[Rumbaugh 91] Rumbaugh, J.; Blaha, M,.; Premerlani, W.; Eddy, F, y Lorensen, W. **Object Oriented Modeling and Design.** Prentice-Hall International Edition. 1991.

[Wilkinson 95] Wilkinson, Nancy M. **Using CRC Cards.** AT & T Bell Laboratories 1995.

[Wirfs Brock 90] Wirfs Brock, Rebecca; Wilkerson, Brian; Wiener, Lauren. **Design Object Oriented Software.** Prentice-Hall Englewood Cliffs 1990.

[Zorman 95] Zorman, Lorna A. The content and composition of scenarios. OOPSLA workshop. Agosto 1995.