Un Proceso para $\mathcal{XR}$ basado en Reglas de Negocio

Maria Carmen Leonardi; Julio Cesar Sampaio do Prado Leite
INTIA- UNCPBA - Tandil, Buenos Aires-Republica Argentina
Dpto. de Informática- PUC-Rio - Rio de Janeiro - RJ Brasil
cleonard@exa.unicen.edu.ar, julio@inf.puc-rio.br

Resumen. Extreme Requirements ($\mathcal{XR}$) es una propuesta que procura dar mayor calidad a la conducción de proyectos en Extreme Programming (*XP*), una de las metodologías más usadas dentro de las metodologías conocidas como ágiles cuyos elementos claves son: poca documentación, simplicidad, análisis como una actividad constante, diseño evolutivo, integraciones y testeo diarios. $\mathcal{XR}$ define una estrategia de requisitos que puede ser acoplada a *XP*, ya que ésta no define una etapa de requisitos, iniciando su proceso con las *stories*, cortas descripciones de lo que debe hacer el sistema y que son utilizadas en todo el desarrollo. En este artículo presentamos un proceso de la familia $\mathcal{XR}$. Nuestro proceso es orientado al cliente, basado en lenguaje natural, de fácil construcción y validación, que ayuda en la comunicación con los clientes definiendo un proceso de requisitos que complementa XR, y que permite convertir al cliente en un miembro del grupo de desarrollo siendo este concepto uno de los pilares de este tipo de metodologías.

1. Introducción

Extreme Programming(*XP*) es una metodología de desarrollo de software eficiente, de bajo riesgo y flexible [1] que cae dentro de las metodologías llamadas "ágiles" cuyas dos principales características son ser adaptativas y orientadas a las personas [4]. Presenta una propuesta incremental de desarrollo, con cortos ciclos de desarrollo y continuo feedback. Está basada en la comunicación oral, testeo y código como estructura de comunicación. Como única documentación de requisitos y utilizada a lo largo de todo el proceso de desarrollo, *XP* utiliza los *User-Stories* como descripciones de lo que debe hacer el sistema, usadas en la etapa de requisitos, diseño y fundamentalmente en el proceso de test. A pesar que en [8] se observa la necesidad de incorporar analistas del negocio para algunos proyectos, no existen trabajos que incorporen modelos explícitos de requisitos a *XP*, salvo la propuesta de [18], en donde muestra la incorporación de *Casos de Uso*[7]. Dada su creciente inserción y utilización dentro de los desarrollos de software orientados a objetos creemos que es importante, desde el área de Ingeniería de Requisitos, adaptar técnicas y estrategias para que puedan ser utilizadas en este tipo de metodologías sin alterar sus principios básicos. En [15] se propone una familia de procesos basada en escenarios llamada

$\mathcal{XR}$ que puede dar mayor calidad a los proyectos en *XP* sin alterar sus principios básicos. Este artículo extiende $\mathcal{XR}$ con la incorporación de un modelo de Reglas de Negocio y un conjunto de heurísticas para derivar las CRCs. De esta forma se presenta una estrategia simple de Requisitos que puede ser utilizada en las primeras etapas de *XP* hasta la obtención de las CRCs.

En *XP* los ingenieros de software se convierten en "auxiliadores "de los clientes[8]. Esto hace que los clientes tengan un rol mas activo en el proceso, trabajando en cercano contacto con el ingeniero [4] para asegurar que se puede obtener una buena calidad de experiencia de negocios en la cual el ingeniero de software podrá confiar y considerar como base para su desarrollo. Por esta razón, creemos que es fundamental que se utilice procesos sustentados en lenguaje natural como base de los procesos $\mathcal{XR}$. Dentro de este contexto se utilizan el modelo de escenarios [14], el LEL [10] y las Reglas del Negocio[13]. El LEL permite representar el lenguaje del Universo de Discurso(*UofD)*[1], acotando el lenguaje externo y enriqueciendo el interno a través de la semántica que se le da a cada término. Unifica el lenguaje permitiendo la comunicación con el cliente. Las Reglas del Negocio son el corazón de cualquier organización ya que gobierna la estructura y el comportamiento de los negocios[6]. Finalmente se propone el modelo de escenarios para representar las *stories* de *XP*, ya que su representación, y su proceso de construcción e inspección permite resolver parte de los problemas de *XP* mencionados en [15]. A partir de estos modelos se propone un conjunto de heurísticas para derivar las CRCs. Si bien se agrega documentación al proceso y *XP* intenta reducir la documentación utilizada, creemos que es útil agregar un modelo inicial para la etapa de requisitos con el objetivo de favorecer la comunicación con el cliente, y mejorar la construcción de las CRCs. Esta documentación se obtiene de forma rápida y fácil, con poco costo. Como es parte de $\mathcal{XR}$ no altera la filosofía de trabajo de *XP* y aporta claridad a las primeras etapas del desarrollo donde la comunicación con el cliente es fundamental para el desarrollo con éxito del software.

El artículo se organiza de la siguiente manera: la Sección 2 introduce brevemente las etapas iniciales de *XP*. La sección 3 presenta los modelos de LEL, escenarios y Reglas de Negocio utilizados en este artículo. En la sección 4 se presenta el proceso para $\mathcal{XR}$ basado en Reglas de Negocio. Un ejemplo correspondiente a un caso real es descrito brevemente en la sección 5. Finalmente la sección 6 presenta conclusiones y futuros trabajos.

2. Una breve descripción de Extreme Programming

Extreme Programming [1][19] es una metodología ágil para equipos de desarrollo de software pequeños o medianos. Es una metodología que se utiliza en organizaciones en donde los requisitos son vagos y variantes. Sus elementos claves son: poca documentación, simplicidad, análisis como una actividad constante, diseño evolutivo, integraciones y testeo diarios. Elimina la gran documentación del sistema como un todo y ataca mínimas partes del sistema que son rápidamente implementadas

[1] "El UofD es el contexto general en el cual el software será desarrollado, operado y mantenido. Incluye todas las fuentes de información y personas o sectores relacionados con la aplicación"[11].

y testeadas. De esta forma el sistema va creciendo junto con el nuevo conocimiento del cliente y sus nuevas necesidades. La figura 1 muestra el proceso (http://www.extremeprogramming.org/map/project.html) general de *XP*.

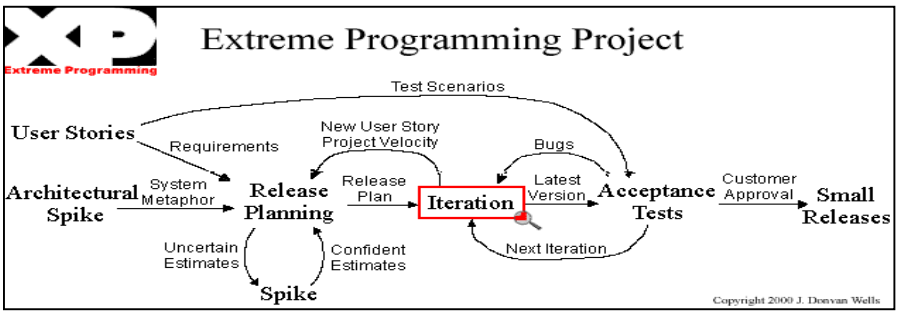

Figura 1. *XP* Project (http://www.extremeprogramming.org/map/project.html)

En este trabajo nos centraremos en las primeras etapas de *XP*, más precisamente en la construcción de *User-Stories y CRCs*. Las *User*-Stories o stories son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son utilizadas como el único documento de requisitos que se genera en *XP*. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Las *stories* guían la construcción de los tests de aceptación, elemento clave en *XP* (deben generarse uno o más tests para verificar que la *story* ha sido correctamente implementada) y son utilizadas para estimar tiempos de desarrollo. En este sentido, sólo proveen detalle suficiente para hacer una estimación razonable del tiempo que llevara implementarla. En el momento de implementar se deben detallar a través de la comunicación con el cliente. Las *stories* sirven también para encontrar las CRCs que serán las futuras clases del sistema. Estas CRCs surgen de una reunión colectiva para determinar las principales CRCs del sistema, sin existir ninguna estrategia para identificarlas. $\mathcal{XR}$ propone utilizar escenarios como representación de las *stories*. Creemos que es posible complementar este enfoque con modelos de requisitos orientados al cliente y un conjunto de heurísticas para derivar CRCs, mejorando la comunicación con el cliente y la construcción de las CRCs sin alterar los principios de simplicidad de la metodología.

3 Modelos de Requisitos basados en Lenguaje Natural

En esta sección se presentan brevemente los modelos utilizados en esta propuesta. Estos modelos se ejemplifican en la sección 5.

3.1. Léxico Extendido del Lenguaje

El Léxico Extendido del Lenguaje, conocido como LEL[10] es una representación de los símbolos que intenta capturar el vocabulario del *UofD*. Su objetivo principal es que el ingeniero de software comprenda el lenguaje que maneja el cliente, entendiendo los términos que utiliza, sin preocuparse por entender el problema. Los símbolos del LEL definen objetos (entidades pasivas), sujetos (entidades activas), frases verbales y estados, cada uno con sus propias heurísticas de construcción[12]. Cada símbolo tiene uno o más nombres que lo identifican, llamados sinónimos. El significado de los símbolos se representa mediante dos tipos de descripciones: nociones e impactos. La noción indica qué es, describiendo la denotación de la palabra o frase. El impacto describe cómo repercute en el *UofD* del problema, es decir su connotación. Estas descripciones se construyen siguiendo dos principios importantes: Maximizar el uso de términos del LEL utilizados (principio de circularidad) y minimizar el uso de símbolos externos al *UofD* (principio de mínimo vocabulario). Mediante estos dos principios se logra un conjunto autocontenido y vinculado de símbolos.

3.2 Modelo de escenarios

Los escenarios son descripciones que evolucionan de situaciones en el entorno. Su objetivo principal es comprender el comportamiento del problema. expresados en lenguaje natural, están conectados con el LEL. Cada escenario tiene la siguiente estructura: Título que lo identifica, Objetivo y un Contexto que describe la ubicación geográfica y temporal del escenario. También se especifican Recursos (medios de soporte, dispositivos que se necesita estén disponibles en el escenario) y Actores (personas o estructuras de organización que tienen un rol en el escenario). Finalmente, los Episodios son una serie ordenada de sentencias, que posibilitan la descripción de comportamiento. Pueden ser opcionales, condicionales o simples, y estar agrupados según su forma de ocurrencia en grupos secuenciales o no secuenciales. Un episodio puede ser descrito como un escenario. Para cada escenario se prevé la descripción de excepciones cuyo tratamiento puede ser realizado por otro escenario. Para algunos componentes del escenario pueden describirse restricciones (requisitos no funcionales). En [14] se describe el proceso global de construcción de escenarios, que derivan un primer conjunto de escenarios basándose en los Sujetos del LEL.

3.3 Modelo de Reglas

Definimos a las reglas de negocios como sentencias sobre la forma en que la empresa realiza negocios[13]. Reflejan las políticas de negocio, cuyas finalidades son: satisfacer los objetivos del negocio, satisfacer los clientes, hacer un buen uso de los recursos, y respetar las leyes o convenciones de la empresa. El modelo de reglas de negocio distingue reglas funcionales y no-funcionales. Existen diferentes patrones[13] para describir cada una de las reglas, en todos ellos las componentes pueden ser entradas en el LEL, conectando naturalmente los modelos. Las reglas no funcionales pueden clasificarse en reglas del macrosistema y reglas de calidad. Las reglas del macrosistema describen políticas que restringen el comportamiento de la organización. Las reglas de calidad son demandas de la organización sobre las

características de sus productos. Usualmente reflejan políticas generales relacionadas a standard o e*xp*ectativas de calidad de la organización. Las reglas funcionales son políticas generales sobre la funcionalidad de la organización. En [17] se presenta una estrategia de definición del modelo de reglas basadas en la semántica y estabilidad de las sentencias que describen a la organización.

4. Un proceso de XR basado en Reglas de Negocio

En esta sección se describe el proceso de XR basado en Reglas de Negocio. En la primer sección se describe globalmente la idea original de XR[15] y la extensión presentada en esta trabajo. Posteriormente se detalla la construcción del modelo de Reglas y el LEL y finalmente la derivación de las CRCs.

4.1 XR para *XP*

Como se menciona en [15], si bien en *XP* no existe un mención explícita de un documento de requisitos, el término requisitos aparece varias veces a lo largo de *XP*. La forma en que *XP* realiza su proceso de definición de requisitos, es a través de la construcción continua de stories, tests, nuevas stories y así sucesivamente, en donde el cliente tiene una activa participación para definir los alcances y prioridades de cada propuesta. XR es una familia de procesos basados en modelos de requisitos orientados al cliente, fácil de construir y comprender, que ayuda tanto a los clientes como a los desarrolladores de *XP*, sin alterar la filosofía básica de *XP*. Si bien genera más documentación, esta es fácil de obtener. XR intenta resolver algunos problemas que presenta *XP* en su forma de manejar los requisitos los cuales se detallan en [15] y son:

1. *XP asume que en la fase "planning game" los negocios son representados por un solo cliente*
2. *Falta de consideración de requisitos no funcionales desde el punto de vista de los negocios*
3. *Falta de links explícitos entre las stories, tareas y código*
4. *Falta de un proceso de producción de tests funcionales*
5. *Falta de un proceso de producción de stories y tareas.*

XR pretende mitigar estos problemas a partir del uso de LEL, escenarios y un proceso simplificado de construcción de los mismos. En este trabajo, se extiende XR con la incorporación de un modelo de reglas, las cuales son orientadas al cliente. Las reglas ayudan en la elaboración de la versiones simplificada de los escenarios, sirven para determinar las prioridades de cada release y son consideradas claves para la evolución del sistema por cambios en los requisitos como resultado de cambios inevitables de las políticas de la organización. También se presenta un conjunto de heurísticas simples para definir las CRCs; estas heurísticas son sencillas y no agregan complejidad a la forma en que *XP* obtiene actualmente las CRS, sirviendo como base para guiar la reunión de diseño de las mismas.

Considerando la filosofía de iteración que propone *XP* a través de las releases, se propone que la construcción de reglas y LEL exceda al proceso de automatización, ya que describe una organización independientemente de la existencia o no de un sistema de software. Con respecto a los escenarios y su rol como representación de *stories*, son mas dependiente de lo que se desea implementar y pueden ser definidos en base a las reglas seleccionadas, ya que el cliente puede determinar prioridades y decidir no definir los escenarios relacionados a las reglas que no le interesa implementar en una determinada release. En base a los escenarios seleccionados, se propone el diseño de las CRCs para cada release.

4.2 Construcción del Modelo de Reglas y de LEL

Como se mencionó en la sección 3, existen estrategias para la construcción de los modelos de LEL y Reglas. En el contexto de la propuesta de XR proponemos una simplificación de la utilización de reglas de negocio y LEL. Para esto se podría utilizar la siguiente estrategia de construcción:

El modelo de reglas puede ser obtenido a partir de entrevistas no estructuradas con los clientes. El cliente básicamente piensa en todos los aspectos de su organización que denotan límites, derechos y responsabilidades de las partes involucradas en el sistema de software que se desea construir: ¿Qué responsabilidades tienen los actores del sistema? ¿Qué límites y derechos? ¿Por qué una actividad se realiza de determinada manera? ¿Existen legislaciones externas que afectan directa o indirectamente su negocio? Estas preguntas pueden ser formuladas por el ingeniero para guiar la conversación. No es necesario que el cliente siga estrictamente los patrones de escritura de reglas, sin embargo, es útil que identifique cual es el término responsable de cada regla y los demás términos involucrados. En esta etapa, aparecen los términos del vocabulario del problema que se está analizando. Se identifican los términos propios de la aplicación (recurso, un proceso, una persona o un sector).[2] De esta forma se crea el LEL para definir el vocabulario, eliminar cualquier tipo de ambigüedad y mal uso de los mismos. El LEL a la vez que facilita la comunicación con el usuario, es una forma natural de agrupar las reglas, por ejemplo, agrupar todas las reglas que tienen a un mismo término como responsable. Por otro lado, como se menciona en XR el LEL es utilizado para obtener un mecanismo de trace, ya que permite que todos los modelos obtenidos en esta etapa y etapas posteriores estén relacionados por el uso de lenguaje restringido, resolviendo el tercer problema presentado en la sección 4.1.

Generalmente por cada regla funcional habrá varias reglas no funcionales asociadas. A partir del conjunto de reglas definido por el cliente, el ingeniero lo completa realizando preguntas que ayudará a relevar los aspectos que no han quedado claros. Por ejemplo, si se escribió una regla funcional, se pueden preguntar los aspectos que las restringen; por el contrario, si se identificaron reglas no-funcionales, en qué momento se aplican.

[2] Se debe tener en cuenta que los escenarios describen principalmente el uso del sistema por esta razón mucho de los términos de LEL estarán relacionados al sistema de software y si son entidades externas importarán las comunicaciones de estas entidades con el sistema de software.

4.3 Construcción de los escenarios del sistema

Como ya se mencionó en la Introducción, $\mathcal{XR}$ propone escenarios para reemplazar las *stories*, y una de las razones es que *XP* no presenta estrategias para la construcción de las mismas. Las *Stories* especifican en un bajo nivel la tarea de una determinada entidad (futuro objeto del sistema), ya que están enfocada a la actividad en esa *story*, perdiéndose el impacto de esa entidad en todo el sistema. Por otro lado, tienen una visión orientada a un único cliente, y no permiten la representación de requisitos no funcionales. Se propone representar a las *stories* con escenarios, enfocando a estos, desde la situación actual del *UofD*, pero con un énfasis en la situación deseada de la presencia de un sistema de software, siguiendo la filosofía de *XP*. De esta forma se amplia el alcance de las stories, que ya no son una descripción de lo que hace el sistema(al igual que los Use-Cases referidos por [18]), sino que pasan a ser una descripción del Universo de Discurso actual pero centrado en la presencia de un software, más adecuada para comprender el problema durante las primeras etapas de desarrollo. Asimismo, los escenarios son inspeccionados por otro tipo de cliente no relacionado directamente con el equipo de desarrollo de proyecto, dando otra visión del negocio y resolviendo el primer problema planteado en la sección 4.1. La representación de escenarios permite definir requisitos no funcionales a través de un campo especial "restricción"que puede ser aplicado a episodios, recursos y contexto, resolviendo el segundo problema. Finalmente, se puede construir los test funcionales a partir de ejemplificar los escenarios, de esta forma un cliente fácilmente construye los test, colocando variantes en cada parte de los escenarios que intentan hacer fracasar los objetivo planteados en los mismos.

El proceso de construcción de los escenarios [14] es simplificado de acuerdo al contexto de *XP* ya que no se desea obtener un modelo exhaustivo de escenarios, sino que, como se indica en $\mathcal{XR}$ se desea obtener un primer conjunto de escenarios que determine una base razonable para producir los tests funcionales y guiar el proceso de diseño. El modelo de Reglas sirve para que el cliente ayude a completar la descripción de los escenarios a partir de las políticas de la organización que desea implementar en el sistema de software.

Se construyen los escenarios a partir de los Sujetos del LEL analizando los impactos para detectar posibles escenarios del sistema. Como se indica en $\mathcal{XR}$, los escenarios son inspeccionados en reuniones, para obtener la aprobación de otro cliente representativo de la organización quien puede utilizar el modelo de Reglas. El modelo de reglas permite tener una visión declarativa y global del negocio, por lo que se puede detectar posibles omisiones en la descripción de los escenarios realizados por un cliente quien se enfocó en alguna actividad en particular olvidando aspectos, que si bien no es el responsable de su ejecución, pueden desencadenarse de su actividad por determinadas políticas actuales de la organización. Este aspecto mitiga el primer problema planteado en 4.1 ya que brinda diferentes aspectos de una actividad. Al tener un modelo de reglas se puede analizar fácilmente y con poco costo, cuan completo es el escenario desde el punto de vista de la organización. Para cada regla funcional asociada a cada escenario se puede determinar si las actividades asociadas a la misma están modeladas correctamente en los escenarios. Asimismo las reglas no funcionales del macrosistema se convierten en restricciones de los

escenarios, mientras que las reglas de calidad son requisitos que abarcan a todo el sistema de software, como se observa en XR.

En [8] se remarca la importancia de la colaboración del analista en la tarea de establecer las prioridades del cliente Esta interacción puede ser realizada a través del modelo de Reglas. Este modelo ayuda al cliente a detectar las prioridades de su negocio, por lo que se puede decidir ya en este nivel retrasar la construcción de un determinado escenario para la próxima sesión. Si ya se han definido varios escenarios, esta decisión selecciona el conjunto de ellos que se utilizará para definir las CRCs. El modelo de escenarios obtenido es utilizado para el diseño de las CRCs así como también para la construcción de los tests funcionales como se explica en XR.

4.4. Construcción de las CRCs

El diseño de las CRCs se realiza en una reunión que llevan a cabo todos los equipos de programación, siguiendo la propuesta de *XP*[1]. En este trabajo se propone un conjunto de heurísticas basadas en [16] para diseñar las CRCs. El formato de las CRCs es el propuesto en [1] a fin de no alterar el modelo de *XP*, sin embargo, si se quiere añadir aspectos de *pre-traceability*[5], sería conveniente seguir el formato propuesto en [16] para determinar el contexto de las responsabilidades, es decir, el conjunto de escenarios en donde cada responsabilidad aparece. Las CRCs son diseñadas en base a los escenarios que han sido seleccionados en cada release.

- SE MODELAN LOS SUJETOS DEL LEL COMO CRCS Y SE DETERMINAN SUS RESPONSABILIDADES A PARTIR DE LOS IMPACTOS. Las clases surgidas a partir de los Sujetos son consideradas clases primarias. Al considerar la presencia del software se pueden realizar dos opciones dependiendo de qué se quiere automatizar:
  - Se define una CRC que modele a este término cuyo comportamiento será el que actualmente hace el término en el mundo real o que registre dicho comportamiento.
  - Los impactos del sujeto se convierten en entradas / salidas del sistema que deben ser manejadas por clases ya existentes. Para esto se analiza quien es el receptor / generador de estos datos a través de los impactos. Es importante el uso de los escenarios para comprender el momento y la forma precisa de la comunicación del sistema con la entidad externa.

Para determinar las responsabilidades, se analiza cada uno de los impactos del LEL correspondientes al término para el cual se ha definido la clase. Al ser clases que han surgido a partir de los Sujetos del LEL, por definición, todos los impactos son acciones que el sujeto realiza, entonces se define al impacto como una responsabilidad del objeto. Existe un caso a tener en cuenta, y es cuando el impacto representa a una frase verbal que ha sido modelada como un término del LEL. Este será analizado en la Identificación de las clases secundarias, pudiendo o no ser modelado como una clase. De todos modos, en principio la responsabilidad queda asignada en esta nueva clase y colaborará con la clase que representa a la acción.

- SE ANALIZAN LOS TÉRMINOS DEL LEL QUE CORRESPONDEN A OBJETOS, ESTADOS Y FRASES VERBALES. Se construye una lista con los términos del LEL que representen objetos, frases verbales y estados. Las clases obtenidas a partir de estos términos son llamadas clases secundarias y son colaboradas de las primarias. Se analiza cada uno de estos términos para determinar si serán modelados o no como objetos. En este trabajo no se tiene en cuenta la parte de las heurísticas relacionadas con los aspectos estructurales de los objetos(atributos y asociaciones), ya que *XP* se centra en definir las *CRCs*, sin considerar los aspectos estáticos de las mismas. Por esta razón las heurísticas analizan la posibilidad de representar a los términos cómo clase, responsabilidades o descartarlos. Un análisis más exhaustivo no caería dentro de la filosofía de *XP*.

  Si el Objeto del LEL tiene comportamiento significativo para el macrosistema se convierte en clase, ya que define una porción de comportamiento importante que puede ser modelado como una abstracción independiente necesaria para que las clases primarias puedan cumplir sus responsabilidades. Para descartarlo como clase se debe analizar si los impactos de los términos no están describiendo acciones similares a alguna clase primitiva, es decir, que si bien es una abstracción propia del sistema, en términos de objetos, no justifica ser una nueva clase. Si el término es una Frase Verbal puede ser modelada como clase o responsabilidad. Es representada como clase cuando tiene características propias que no pueden ser asignadas a otras clases. Si este es el caso se modela como objeto, sino, se convierte en una o más responsabilidades de los objetos involucrados. Finalmente los Estados, por definición, afectan a algunas componentes del sistema que han sido modeladas como términos del LEL. Si el término del LEL al cual afectan fue modelado como clase, se analiza primero si el estado no puede incorporase a la CRC para modelar el nuevo estado, pero si tiene impactos que pueden ser modelados como responsabilidades independientes, entonces es candidato a ser modelado como una clase diferente. Para cualquiera de estos términos, si existen varias reglas no funcionales asociadas, será conveniente modelarlos como clases para poder especificar las reglas.

  Se determina las responsabilidades analizando, para cada clase definida, los impactos del término del LEL correspondiente. Es necesario tener en cuenta la categoría del término para el cual se está determinando las responsabilidades ya que, según sea la categoría, cambia el sentido de los impactos pudiendo, en algunos casos, agregar responsabilidades a otras clases.

- SE VALIDAN LAS CRCS A PARTIR DE LOS ESCENARIOS. Las CRCs resultantes son evaluadas de una forma similar a la propuesta en [3]. Se toma cada *escenario* y se ejecuta determinando si las CRCs involucradas y sus responsabilidades pueden llevar a cabo las acciones descriptas en el mismo. De esta forma se pueden refinar las responsabilidades agregando la funcionalidad necesaria para satisfacer cada escenario. Como los escenarios han sido construidos teniendo en cuenta las reglas, las CRCs resultantes de esta validación contendrán a través de sus responsabilidades, toda la funcionalidad definida por el modelo de reglas.

Es importante resaltar que los pasos detallados anteriormente son una guía. En el contexto de *XP*, se podría argumentar que se genera un gran volumen de documentación, pero como se destaca en [9] más importante que la documentación,

es el proceso de escribirla que permitirá tanto al cliente como al ingeniero obtener un conocimiento mayor del problema para el cual se va a construir software. En este sentido, el modelo de reglas sirve para guiar en la construcción de los escenarios permitiendo que el cliente piense en términos de sus negocios, todo esto acompañado de un lenguaje uniforme modelado en el LEL. A su vez las heurísticas de definición de CRCs completan esta estrategia guiando de una forma sencilla la construcción de CRCs. De esta forma se construyen los escenarios que representaran a las *stories* y se definen las CRCs de forma tal que reflejen el comportamiento, políticas y lenguaje de la organización.

5 Caso de Estudio

El Establecimiento Don Juan[3] S.A es una PYME de Explotación Agrícola / Ganadera y Transportes Generales. Consta de dos socios mayoritarios. Es una empresa mediana con una estructura de empleados permanentes (4 empleados en el área de Comercio, 6 empleados en el área Rural y 7 choferes de Transportes Generales) y Personal Rural transitorio (variable según temporada de producción) con un máximo de 25. Sus funciones principales son: Producción Agrícola / Ganadera, Comercialización de la producción propia y de terceros, y Transportes Generales. En este ejemplo se muestra el análisis de una parte del área de comercialización del producto principal de la e*xp*lotación que es el cultivo de papa. El análisis está basado en entrevistas realizadas a uno de los principales empleados del área de Comercio. A continuación se muestran algunas de las entradas del LEL correspondiente a diferentes entidades de la organización.

Planta Procesadora / Planta
Nociones
- Cliente de Don Juan que procesa la papa comprada obteniendo papas bastones prefritas congeladas de máxima calidad.

Impactos
- Realiza contratos con Don Juan para establecer los requisitos y condiciones para la adquisición de cada variedad del producto.
- Recepciona el envío del producto en uno o más viajes
- Toma muestras de cada carga de los camiones enviados por Don Juan
- Puede rechazar alguno o todos los camiones de una recepción según el grado de calidad establecido y estipulado en el/los contrato/s
- Realiza descuentos y bonificaciones según el grado de calidad de los productos enviados
- Define el grado de calidad de cada año.

Contrato /contrato de Adquisición y producción de papas
Nociones

[3] El ejemplo de esta Sección se basa en documentación perteneciente a Don Juan S.A. y a contratos realizados por este establecimiento con una importante Planta Procesadora de Papas establecidas en la zona.

- Es un documento firmado por un responsable de la Planta y uno por Don Juan que establece las condiciones de producción, compra, plazos de entrega y pago del producto.
- Se debe hacer uno por cada variedad comercializada y por año.
- Contiene las condiciones de entrega del producto y el procedimiento de clasificación
- Contiene la definición de grado de calidad válido parase contrato
- Describe el precio a pagar y bonificaciones y descuentos

Impactos

- Es utilizado como base para el rechazo o aceptación del producto
- Es utilizado como base en caso de no cumplimento de los pagos
- Es utilizado como base en caso de no cumplimento de Don Juan

Papa / producto

Nociones

- Es el cultivo producido, comprado y vendido por el establecimiento Don Juan
- Es de una determinada variedad
- Tiene un peso y un tamaño
- Tiene una época de siembra y otra de cosecha

Impactos

- Es sembrada y posteriormente cosechada
- Se venden por toneladas o bolsas según sea el tipo de cliente que compre
- Puede ser refrigerada en cámaras frigoríficas propias o alquiladas
- Puede ser lavada y cepillada antes de comercializarse
- Deben cumplir un determinado grado de calidad
- Es transportada a los distintos clientes en camiones

Algunas de las Reglas obtenidas a partir de las entrevistas son las siguientes:

1. Don Juan vende su propia producción de papas y producción comprada a terceros que no realizan comercialización.
2. Don Juan vende a Mayoristas, Consignatarios y Plantas.
3. Para cada tipo de cliente Don Juan tiene sus propias políticas de comercialización.
4. Según el estado financiero, el stock de producto o conveniencia del mercado, Don Juan sale a vender(oferta) a Mayoristas o Consignatarios, eligiéndolos por características comerciales
5. Ante una demanda se analiza si conviene satisfacerla por precios de mercado y por características comerciales
6. Si no es posible satisfacer demandas de varios clientes, se prioriza por características comerciales.
7. El sistema de pago modalidad Cuenta Corriente es para los clientes que pertenecen al pool de clientes.
8. Tanto para Mayoristas como para Consignatarios se manejan los siguientes mínimos de venta: Interior de la provincia de Buenos Aires: la carga correspondiente a un camión. Resto del país: la carga correspondiente a un

equipo completo Existen excepciones que se manejan según estado financiero o características comerciales.

9. Los pedidos de las Plantas se hacen por tonelada del producto.
10. Al inicio de cada siembra las Plantas envían un e*xp*erto agrónomo para relevar las características del campo. Si el relevamiento es aprobado se comienza con la elaboración de un contrato.
11. Don Juan entrega siempre la cantidad de producto establecida en el contrato, si no llega con producción propia, compra a terceros.
12. Si la Planta rechaza camiones, Don Juan cumple con el contrato con mercadería comprada a terceros o propia.
13. El precio convenido con la Planta en cada contrato incluye el valor del flete a destino.
14. La Planta toma 3 muestras de cada camión.
15. La Planta acepta o rechaza el producto según el grado de calidad que se obtuvo en la muestra.
16. La Planta realiza descuentos y bonificaciones sobre el precio establecido en el contrato a partir del resultado de las muestras.
17. Don Juan utiliza el resultado de las muestras realizadas por la Planta para tomar decisiones en cuanto a la de producción de papa y a la selección a realizar en sus próximos envío.
18. Cada negociación con los Mayoristas y Consignatarios es independiente, dependiendo de las características comerciales, estado financiero y disponibilidad del producto.
19. Con el viaje enviado a Mayoristas y Consignatarios se adjunta la guía frutihortícola y eventualmente la factura si el cliente lo solícita.
20. El valor del flete del viaje enviado a Mayoristas y Consignatarios es pagado por el cliente al llegar a destino.
21. Solo si el chofer es empleado de Don Juan el cliente puede hacer pagos de la mercadería que se transporta en el viaje.
22. Si un Mayorista o Consignatario rechaza un camión Don Juan intenta renegociar, se vende a otro cliente dentro del recorrido del viaje o finalmente se descarta.

A partir del término del LEL Planta Procesadora y a las Reglas de Negocio relacionadas con ella (Reglas 9-17) se pueden construir diferentes escenarios. En este ejemplo mostramos un escenario relacionado a los aspectos que debe almacenar el sistema de software con respecto a la entrega del producto en la Planta:

NOMBRE: Entrega de productos a Planta
OBJETIVO: Don Juan desea registrar la entrega de productos a la Planta y todas las acciones que ésta realiza en la recepción.
CONTEXTO: Ocurre en la Planta en fecha y forma estipulada en contrato
ACTORES: Don Juan, Planta
RECURSOS: Productos, camión, muestra, grado de calidad, contrato
EPISODIOS

Don Juan envía en camiones la entrega de los productos a la Planta.
La Planta toma tres muestras de cada camión

La Planta acepta o rechaza del camión en base al grado de calidad

Si la entrega es aceptada, se registran los descuentos y bonificaciones realizados por la Planta según haya sido el resultado de la muestra para evaluar el precio final.

Si se rechazó algún camión, Don Juan debe enviar más producto para cumplir el contrato, el cual puede ser propio o comprado para llegar a la cantidad establecida.

Otro escenario relacionado con los viajes enviados a Mayoristas o Consignatarios y relacionado con las Reglas 19-22, puede ser:

NOMBRE: Envío de fletes a Mayoristas o Consignatarios
OBJETIVO: don Juan envía el flete al cliente que puede ser un Mayorista o Consignatario
CONTEXTO: la venta ya ha sido acordada entre Don Juan y el cliente
ACTORES: cliente Don Juan
RECURSOS: viaje, producto, guía frutihortícola, factura, chofer, flete
EPISODIOS

Don Juan envía el viaje con el producto, enviando también la guía frutihortícola y la factura correspondiente si el cliente la solicita

El flete es pagado al llegar el mismo a destino.

Si el chofer del flete es empleado de Don Juan, el cliente puede hacer pagos al llegar el viaje, quedando registrado en el sistema.

Si el cliente rechaza el pedido se intenta renegociar.

Si no acepta se intenta ubicar el producto entre los clientes que Don Juan tiene en la zona o traer de vuelta el producto para descarte.

Ambos escenarios reflejan las entregas del producto a diferentes tipos clientes que trabajan cn Don Juan, cada uno con sus propias políticas de entrega y rechazo. El siguiente paso es utilizar las reglas para decidir que conjunto de escenarios se implementará en cada release. De las entrevistas se determina que todos los aspectos relacionados con la Planta son los que Don Juan desea priorizar debido a la importancia del cliente y a sus exigencias. A partir del LEL, y el modelo de escenarios seleccionados se diseñan las CRCs utilizando las heurísticas presentadas en la sección 4. El modelo de reglas no es utilizado directamente en esta etapa ya que al ser usado como fuente para construir los escenarios, se puede considerar que éstos últimos reflejan las políticas de la organización. La figura 2 muestra las CRC correspondiente a Planta. Al modelar al software, las responsabilidades de esta CRC es almacenar las acciones realizadas por la Planta y que Don Juan tiene interés en registrar.

nombre: Planta	
Responsabilidades	Colaboradores
• RealizarContratos	grado de calidad
• RecepcionarEnvio	viaje, chofer
• TomarMuestas	muestra
• RechazarCamión	muestra, grado de calidad
• RelizarDescuentosBonificaciones	contrato, grado de calidad,
• DefinirGradoCalidad	grado de calidad

Figura 2. CRC correspondiente a una Planta Procesadora

La siguiente CRC corresponde a una Muestra. Si bien la muestra la realiza la Planta, se define una clase que la refleje, ya que, como se define en las reglas, es interés de Don Juan seguir un control de cada muestra tomada basándose en las características de estos productos y en el grado de calidad establecidos por la Planta. Además se sugirió registrar muestras propias de su producción y de la comprada a los productores para cumplir con los requisitos y hacer análisis estadísticos a través de diferentes temporadas.

Nombre: Muestra	
Responsabilidades	Colaboradores
Origen	productor
DarPeso	Planta
DarDefectosProducto	Planta
DarPorcentajeNoCumplimientoDefecto	grado de calidad
DarMalformaciones	Planta
DarPorcentajeMalformaciones	grado de calidad
.....	

Figura 3. CRC correspondiente a una Muestra de la Planta Procesadora

Este ejemplo sirve para mostrar como la utilización de la estrategia presentada en este trabajo puede implementar un proceso $\mathcal{XR}$ en un caso de estudio donde si bien la funcionalidad no es compleja maneja un conjunto importante de políticas de negocio según el tipo de clientes y que pueden variar por diversos motivos internos y externos a la organización. Dentro de *XP*, nuestra propuesta se enfoca en la parte de "planning game", por lo tanto, sobre la base de las reglas y escenarios se deben estimar los costos y determinar finalmente las prioridades para empezar con la etapa de "scoping". Poder automatizar estas políticas permite que Don Juan posea información que le permitirá tomar decisiones y planificar futuras acciones con respecto a su interacción con las Plantas Procesadoras, principales clientes de este establecimiento.

6. Conclusiones

En esta propuesta se presenta un prototipo de proceso para $\mathcal{XR}$. El modelo de LEL unifica y permite la comunicación con el cliente. Asimismo el LEL es usado como un mecanismo de trace que relaciona todos estos modelos así como las CRCs y el código a través del uso de un lenguaje restringido. El modelo de Reglas describe el

conocimiento de una forma sencilla y cercana a la forma en que el cliente percibe a la organización. El modelo de reglas también ayuda a identificar prioridades, siendo esto de gran utilidad para determinar en cada release cuales serán los escenarios a implementar ya que el cliente puede decidir que los procedimientos asociados a determinadas reglas pueden ser dejados para próximas releases. La representación de las stories mediante escenarios resuelve los problemas planteados en [15] con relación a las *stories* mediante su representación y su proceso simplificado de construcción y de inspección. La estrategia define también un conjunto de heurísticas simples para construir las CRCs a partir de los modelos de requisitos.

Los modelos utilizados favorecen la visión adaptativa, orientada al cliente e incremental de la metodología. En particular, el modelo de reglas, orientado a la organización, permite identificar fácilmente los cambios y que ésta se adapte fácilmente cuando las reglas cambien [2]. Un cambio en la organización puede afectar a las reglas, invalidándolas o generando nuevas. El cliente percibe fácilmente los cambios y cómo repercuten en las reglas, ya que éstas están descriptas en su propio lenguaje. A partir de ahí se reformulan los escenarios relacionadas a las reglas y se modifican las CRCs involucradas.

La estrategia presentada en este trabajo es un prototipo de un proceso dentro de las posibles familias de procesos de $\mathcal{XR}$. Como todo prototipo se encuentra en continuo estudio y desarrollo. Para esto es necesario aplicarlo en diversos casos de estudio orientados al negocio. Dada una importante tendencia dentro de la comunidad de objetos a utilizar este tipo de metodologías[4] y a recientes trabajos que estudian integrar a XP con Requisitos (por ejemplo el XP2001 Workshop on Requirement Negotiations[www.XP2001.org], y los Workshops de Metodologías Ágiles presentados en OOPSLA'01[www.oopsla.org]), se pretende profundizar la incorporación de modelos de requisitos simples que se adapten fácilmente a las mismas y que permitan describir a la organización de una forma ágil y adaptable a los cambios inevitables y necesarios de cualquier organización.

Referencias

[1] Beck Kent "Extreme programming explained: embrace change", Addison-Wesley, 2000.

[2] Blaze Software "Architecture for change: The Rule Engine Proposition" www.blazesoft.com

[3] Cockburn Alistair "Responsibility-based Modeling", Technical Memo HaT TR-99.02. http://members/aol.com/humansandt/techniques/responsabilities.htm.

[4] Fowler Martin "New Methodology", http://martinfowler.com/articles/newMethodology.html

[5] Gotel O, Finkelstein A. "An analysis of the Requirements Traceability Problem", International Conference on Requirements Engineering,1994, pp.94-101.

[6] Gottesdiener "Business RULES Show, Power, Promise", Application Development Trends, Vol 4, nro. 3 1997 http://www.ebgconsulting.com/

[7] Jacobson I Booch G, Rumbaugh J. "The Unified Software Development Process", Addison Wesley- 1999.

[8] Jeffries Ron "Business Analysis in Extreme Programming", *XP* Magazine- http://www.*XP*rogramming.com/*XP*mag/. 09/01/2000.

[9] Lawrence " Designers Must Do the Modelling", IEEE Software Vol 15 Number 2, March/April 1998 Pp. 31-33.

[10] Leite J.C.S.P, Franco A "O Uso de Hipertexto na Elicitaçao de Linguagens da Aplicaçao", Anais de IV Simpósio Brasileiro de Engenheria de Software, octubre de 1990. pp.134-149.
[11] Leite J.C.S.P, A. Pádua Albuquerque Oliveira. "A Client Oriented Requirements Baseline", Proceedings of the Second IEEE International Symposium on Requirements Engineering , IEEE Computer Society Press, 1995, pp.108-115.
[12] J.C.S Leite. "Ingeniería de Requisitos", Notas de Aula, Facultad de Cs. Exactas, UNCPBA, Tandil, Argentina, noviembre de 1997.
[13] Leite J.C.S.P, Leonardi Ma. Carmen, "Business rules as organisational Policies", IEEE IWSSD9: Ninth International Workshop on Software Specification and Design, IEEE Computer Society Press, 1998, pp. 68-76.
[14] Leite J, Hadad G, Doorn J, Kaplan G., "A Scenario Construction Process" Requirements Engineering Journal, Springer-Verlag, 2000.Vol.5 N1, pp:38-61.
[15]Leite J.C.S.P "Extreme Requirements (XR)", Jornadas de Ingeniería de Requisitos Aplicada. Sevilla, 11 y 12 de Junio del 2001.
[16] Leonardi Carmen, Maiorana Vanesa , Balaguer Federico, "Una Estrategia de Análisis Orientada a Objetos Basada en Escenarios", Actas de II Jornadas de Ingeniería de Software, JIS97, Dpto. de Informática, Universidad del país vasco, San Sebastián, España, 1997, pp. 87-100.
[17] Leonardi Carmen, Leite J.C.S.P, Rossi G., *"Estrategias para la identificación de Reglas de Negocio",* Proceeding de Sbes98 "Simposio Brasilero de Engenharia de Software" Sociedad Brasilera de Computacao, Brasil, 14-16 de Octubre de 1998. Pp. 53-67.
[18] McBreen Pete, "Incremental Requirements Capture", *XP* Magazine-http://www.*XP*rogramming.com/*XP*mag/. 09/01/1999.
[19] Extreme Programming: A gentle introduction. http://www.extremeprogramming.org/