Universidad Tecnológica Nacional

Facultad Regional Córdoba

Dirección de Posgrado

Especialización en Ingeniería en Sistemas de Información

Gestión de Procesos de Desarrollo de Software con Herramientas ALM

José Mauricio Silclir

D.N.I.:31121991

Email: msilclir@gmail.com

Córdoba, Abril de 2013

Índice

Índice de Figuras

Índice de Tablas

No table of figures entries found.

Resumen

El contar con un proceso para el desarrollo de software a nivel organizacional se ha convertido en una práctica clave para entregar productos con cada vez mejor calidad, y a la vez impulsar la mejora continua de los equipos y la organización en sí. Cuando los procesos de desarrollo se volvieron lo suficientemente complejos, se comenzaron a desarrollar herramientas que ayuden en su definición, instanciación y automatización. Estas herramientas son lo suficientemente flexibles para adaptarse a cualquier proceso, pero esta flexibilidad incrementa a la vez la complejidad a la hora de configurarlas para su utilización en un proyecto y producto dados. Este trabajo muestra cómo, en base a una taxonomía dada, se puede configurar automáticamente una herramienta de gestión de acuerdo a las características propias de un producto y proyecto, utilizando RTC como herramienta de gestión, y Scrum como metodología de desarrollo.

Palabras claves: Taxonomía, Metodologías Agiles, Scrum, Rational Team Concert, Herramientas de Gestión de Ciclo de Vida, Procesos de Desarrollo de Software

Abstract

<abstract in english>.

Keywords: Agile, Scrum, Tools, software development process

1. Introducción

Las metodologías ágiles han adquirido un protagonismo particular en la industria del software, ofreciendo tanto un conjunto de prácticas de ingeniería para el desarrollo de software como algunos marcos de trabajo (frameworks) de gestión de proyectos [Glazer, y otros, 2008], siempre bajo la filosofía ágil guiada por el Manifiesto Ágil [Beck, y otros, 2001].

Como consecuencia, han surgido diversas herramientas tales como Rational Team Concert de IBM[1] o Team Foundation Server de Microsoft[2], cuyo objetivo es dar soporte y mejorar la productividad de los equipos, tanto a nivel del desarrollo y prueba de las aplicaciones como así también de la gestión de proyectos y entregas.

Sin embargo, dada la inherente orientación de las metodologías ágiles a la aplicación práctica de las distintas técnicas de desarrollo y gestión de software, se requiere un conocimiento acabado de las prácticas de software, bases de las metodologías ágiles (manifiesto), el marco de trabajo particular que se utilizará para la implementación de estas metodologías y las herramientas seleccionadas para desarrollar y gestionar un proyecto de software en particular.

De esta manera, este trabajo de especialidad pretende proporcionar un conocimiento acabado de una de las metodologías ágiles, Scrum, y de una de las ofertas de herramientas para desarrollo de software, Rational Team Concert (RTC). Con esta información, se desarrollará una prueba piloto de una aplicación que permita unir estos conocimientos para configurar la herramienta (RTC) de acuerdo a las características particulares de cada proyecto de desarrollo de software.

Cabe destacar que este trabajo es una parte del proyecto de tesis “Una taxonomía de procesos para la configuración de proyectos ágiles basados en Scrum”. Dicho trabajo de tesis es el que finalmente proporcionará una taxonomía que unirá los dos conjuntos de conocimientos mencionados anteriormente (conocimientos de Scrum y conocimientos de configuración y uso de RTC) y servirá como entrada principal a la aplicación para configuración de RTC, a desarrollarse como parte del presente proyecto de especialidad.

2. Contexto y Fundamentación

[1] https://jazz.net/products/rational-team-concert/

[2] http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

a. ¿Por qué es importante contar con un proceso definido a nivel organizacional?

Si bien suelen haber necesidades particulares para cada proyecto en términos de procesos de desarrollo, existe también un número de razones para contar con un marco de trabajo de procesos estándar en la organización:

- La estandarización de procesos es clave para dar paso al entrenamiento, gestión, revisión y soporte de herramientas;
- Con métodos estándares, la experiencia de cada proyecto puede contribuir a la continua mejora de procesos en la organización;
- La estandarización de procesos provee la estructura básica para la medición;
- Dado que la definición de procesos conlleva tiempo y esfuerzo, es impráctico pro-ducir nuevos procesos para cada proyecto;
- Como las tareas básicas son comunes a todos los proyectos de desarrollo, un proceso definido estándar necesitará solamente de algunas pocas modificaciones para cubrir los requisitos particulares del proyecto [Humphrey, 1989].

Partiendo de estas ideas básicas, a lo largo de la historia de la industria del software se han identificado una importante cantidad de grandes ideas y conocimientos disponibles acerca de cómo desarrollar efectivamente software, partiendo de la programación estructurada tradicional [O.J. Dahl, 1972] hasta llegar a las actuales tecnologías de desarrollo [J.L.Pfleeger, 2002].

La mayoría de esas propuestas partieron de la premisa principal de que la calidad de un producto está influenciada por la calidad del proceso que se utiliza para construirlo [Humphrey, 1989] [Sommerville, 2010]. Esto determina la necesidad de que todo proyecto de software cuente con un proceso de desarrollo robusto [Royce, 1970] [Barry W. Boehm et al., 1984].

Con estas ideas presentes, se puede entrever la generación de un nuevo requisito para los equipos de desarrollo: no sólo necesitan definir y tener acceso a información detallada sobre tecnologías de desarrollo específicas, incluyendo lenguajes de programación y de bases de datos y diversas herramientas y ambientes de desarrollo, sino que además deben contar con la habilidad de definir y acceder al proceso de desarrollo bajo el cual se llevarán a cabo las actividades del proyecto [Humphrey, y otros, 1989]. Dicho proceso incluirá las mejores prácticas de desarrollo, tales como metodologías ágiles, modelos iterativos, y desarrollo de software dirigido por el riesgo y la calidad

[OMG, 2008], y proveerá a los profesionales el marco de trabajo que necesitan para llevar a cabo un trabajo profesional de manera consistente [Humphrey, y otros, 1989].

Bajo estas premisas, se presentan algunos problemas al pensar cómo definir el proceso de desarrollo en una organización [OMG, 2008]:

1. Los miembros del equipo no tienen un acceso fácil y centralizado al mismo cuerpo de información para la ejecución del proceso cuando lo necesitan.
2. Se deben combinar e integrar contenidos y procesos de desarrollo que están disponibles en formato propietario, junto con las diferencias en los estilos y formas de presentación de cada uno de ellos.
3. Cada organización debe definir un enfoque sistemático y organizado que sea apropiado para sus necesidades.

Cada organización que desarrolla software tiene definido un proceso de desarrollo, basado generalmente en alguna metodología estándar de la industria [J.L.Pfleeger, 2002], y para su implementación dichas organizaciones desarrollan sus propios modelos de proceso, que guían y proporcionan soporte a los desarrolladores de software estableciendo qué actividades y pasos deben ejecutar para la producción de software de calidad [Rolland, 1995] [Scacchi, 2002]. Un lenguaje de modelado de proceso amigable y no ambiguo y una herramienta que soporte dicho lenguaje son elementos muy importantes para las organizaciones al momento de definir, mantener, verificar y validar los procesos [Oliveira, 2010].

b. ¿Por qué es importante tener un proceso automatizado?

Paralelamente a la necesidad de definir procesos de desarrollo, en las organizaciones surge la necesidad de disponer de herramientas que faciliten el modelado de dichos procesos, y su automatización.

De acuerdo por lo descripto por Humphrey [Humphrey, 1989], la razón principal detrás de la automatización de procesos es la necesidad de aumentar la calidad y productividad del trabajo en los equipos. Mediante la automatización de ciertas tareas, se ahorra esfuerzo y se eliminan posibles errores humanos.

Sin embargo, en ese mismo artículo, Humphrey establece que para considerar la automatización de procesos como una manera de mejorar la eficiencia del mismo, el proceso de desarrollo de una organización y/o equipo debe ser estable y razonable-mente efectivo. Si la organización no ha implementado un proceso con esas características, la instalación de cualquier herramienta puede ser traumática.

De esta manera, como un paso previo, la organización necesita delinear una estrategia de automatización que guíe el uso de nuevos métodos y técnicas.

Entonces, las herramientas de automatización de procesos no sólo deben facilitar el modelado de procesos, sino también automatizar todo o parte de la ejecución de los mismos. Con este propósito, se pueden identificar una serie de requerimientos básicos que aseguren que una herramienta de modelado de procesos dada contribuya con la mejora en la calidad y la productividad del trabajo realizado [Humphrey, 1989]. Hoy en día, hay múltiples herramientas para el modelado de procesos disponibles en el mercado, con distintas características que las distinguen una de otra. El principal punto estará en evaluar los requisitos que específicos de un proyecto, producto u organización, y en base a esos requerimientos decidir qué herramienta es la más adecuada.

c. ¿Por qué Rational Team Concert?

Para poder efectivizar todas las ventajas ofrecidas por la automatización de procesos, es necesario contar con una herramienta adecuada, que nos permita definir las tareas adecuadas y automatizar buena parte de la ejecución de las mismas.

Con el fin de acotar el alcance de manera de obtener resultados más concretos, el presente trabajo parte de la base de que una compañía ya cuenta con un proceso definido a nivel organizacional, basado en el marco de trabajo de Scrum.

Por otra parte, también se asume que tanto la definición como la instanciación de dicho proceso se realizan a través del uso de una o más herramientas de modelado y ejecución de procesos.

Un tercer aspecto que se debe considerar es que la presente tesis se desarrollará como parte de las tareas del Laboratorio de Investigación y Desarrollo en Ingeniería y Calidad del Software (LIDICALSO). El mismo presenta una serie de requisitos que deben cumplimentarse en el marco de los desarrollos llevados a cabo en el laboratorio.

En el informe titulado “Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software” [Silclir, y otros, 2010] se evalúan las características básicas (requerimientos) que deben considerarse al seleccionar una herramienta de modelado de procesos de software. En dicho trabajo, se toman como base una serie de requisitos planteados para el trabajo en el laboratorio de investigación LIDICALSO y se evalúan, a modo de prueba piloto del método, tres herramientas, siendo RTC la más adecuada de acuerdo a los resultados de la aplicación del método. Siendo que este trabajo se realiza dentro del marco del mismo laboratorio de investigación y dentro de la misma línea de investigación, la conclusión arribada en esa publicación es válida también en este contexto.

d. Una taxonomía para la configuración de los procesos en RTC

La necesidad de un proceso para el desarrollo de software ha sido planteada desde los inicios mismos de la actividad, medio siglo atrás. A partir de entonces se han diseñado procesos que ayudaran a los equipos a desarrollar software de una manera cada vez más productiva.

Cuando los procesos de desarrollo se volvieron lo suficientemente complejos, se comenzaron a desarrollar herramientas que ayuden en su definición e instanciación. Algunas de ellas son

específicas para ciertas actividades y prácticas del software, y otras fueron diseñadas para cubrir la mayoría (sino todas) las actividades. Así, se llegó a lo que hoy conocemos como Life Cycle Management Tools: Herramientas de Gestión del Ciclo de Vida (de un producto).

Estas herramientas ofrecen en general dos grandes funcionalidades. Primero, la de definición del proceso a utilizarse, especificando las entidades principales, entradas, salidas y flujos de trabajo. Y en segundo lugar, la instanciación de dicho proceso para un producto/proyecto específico. Un ejemplo claro de este tipo de herramientas es RTC.

Es claro que una de las grandes ventajas que ofrecen este tipo de herramientas, y en particular RTC, es la flexibilidad al momento de la definición de los procesos de desarrollo y la posterior automatización de ciertas actividades del mismo. Esta es una característica absolutamente necesaria para poder adaptarse a los distintos tipos de productos, con sus realidades particulares (entorno, cultura, mercado, etc.).

Pero es justamente en esa flexibilidad donde surge una nueva incógnita: dado un producto particular, ¿cuál es la configuración más adecuada?

Llegado a este punto se presenta la hipótesis de que, dado el conocimiento de las características de un producto y un proyecto, existe una configuración óptima del proceso para ese producto en RTC utilizando Scrum como metodología de desarrollo. En otras palabras, la pregunta específica a responder es: ¿cuál es la configuración adecuada del proceso en RTC para un producto en particular, utilizando como base el marco de trabajo Scrum?

Esta cuestión obliga a dar un paso hacia atrás y pensar cómo se configuran los proyectos al día de hoy. Y la respuesta básica es que dicho conocimiento se encuentra centrado en unas pocas personas, con experiencia en el uso de la herramienta (RTC), en la definición de procesos (basados en Scrum) y que entienden además los detalles de la gestión de un producto o entrega dada.

Es decir, cada vez que se va a iniciar un nuevo producto o proyecto de cualquier tipo, el procedimiento implica contactar a uno de los expertos para que evalúe la situación del producto, haga un análisis de sus requisitos principales en términos de gestión de backlog[3], entregas comprometidas, cantidad de equipos, y otras muchas variables, y determine la configuración que pareciera ser óptima en base a su propia experiencia.

[3] Repositorio de ítems pendientes de ser completados.

Es de notar también que, si bien el conocimiento está centrado en pocas personas, incluso ninguna de ellas es experta en todos los temas que se mencionaron como necesarios a la hora de determinar una configuración del proceso. Muchas veces es necesario el trabajo en conjunto para lograr el resultado más adecuado a las características del producto y a la realidad de la organización.

De esta manera, ¿cuáles son, si existen, las alternativas de solución que se pueden diagramar?

La primer idea que parece surgir es la de contar con una plantilla de proceso previamente definida en RTC. En este momento, las metodologías y ciclos de vida de desarrollo a utilizar están definidas para la organización, con lo cual solamente deberíamos definir una plantilla para cada metodología y ciclo de vida de desarrollo. Así, ante un nuevo producto, solamente se debería determinar el tipo de proceso a seguir y aplicar la plantilla correspondiente en RTC.

Una solución similar a esta es la que se utiliza en Motorola Mobility of Argentina. Dicha organización cuenta con un proceso definido a nivel organizacional, tomando como base la metodología Scrum. Para cada uno de los productos en desarrollo, se crea una instancia del mismo en RTC, utilizando una plantilla previamente definida. Dicha plantilla ofrece las funcionalidades típicamente necesarias para cualquier producto desarrollado en la organización.

Sin embargo, el tipo de metodología de desarrollo o ciclo de vida no es suficiente para poder generar el proceso adecuado en RTC. Hay características que van más allá de la metodología que hacen que cada producto en particular sea totalmente particular y distinto. Entre esas características se encuentran:

- Cantidad de equipos trabajando en el producto
- Número de versiones del producto que se mantienen en paralelo
- Cantidad de entregas planificados en paralelo
- Condiciones contractuales de entrega del producto
- Integración (o no) de los equipos de prueba de sistema
- Otras…

Para el caso de Motorola Mobility of Argentina, los responsables de la gestión de los procesos en RTC se han encontrado en la situación de tener que modificar las especificaciones del proceso para cada producto que se ha ido creando, ya que cada uno de estos presenta variaciones en las características listadas.

La segunda alternativa que asoma es la de generar una plantilla de proceso para cada una de las características anteriores. Sin embargo, esta opción es muy difícil de implementar y mantener dado que el número de características es alto e incierto.

Por lo tanto, asoma una tercera opción: crear una plantilla de proceso para cada proyecto en particular, en base a sus características.

Así, se divide el problema en dos etapas principales:

1. Determinar las características principales de un proyecto en particular
2. Generar la plantilla de proceso en RTC de acuerdo a las características obtenidas en el paso anterior

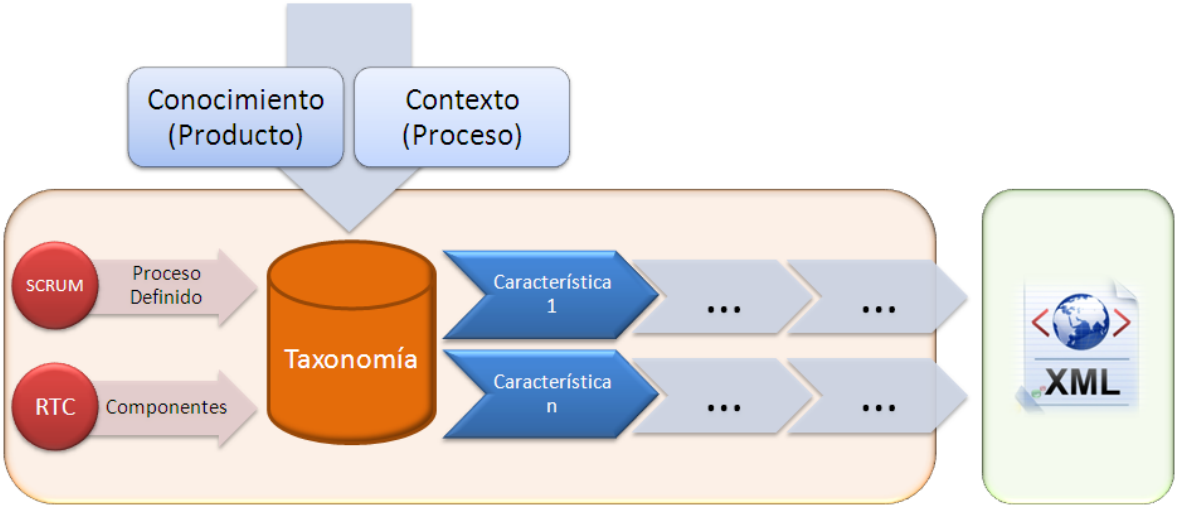


Figura 1: Esquema de generación de una taxonomía de procesos

El presente trabajo describe una taxonomía de configuración del proceso de desarrollo y gestión del producto en una herramienta tomando como información de entrada el conocimiento del producto, representada en la Figura 1: Esquema de generación de una taxonomía de procesos.

3. Métodos y materiales: Prueba de concepto de la Taxonomía

La implementación de la taxonomía de configuración de procesos mencionada en la sección anterior involucra las siguientes etapas:

1. Determinar cuáles son las posibles situaciones o problemas a la hora de estructurar un proyecto de desarrollo de software para gestionarlo utilizando Scrum.
2. Determinar cuáles son las respuestas a esas situaciones o problemas que las metodologías ágiles / Scrum ofrecen.
3. Determinar cuáles son las posibles implementaciones a esas respuestas metodológicas en la herramienta ALM.

Los puntos 1 y 2 de la lista anterior serán resueltos con la taxonomía de procesos, a desarrollar como parte del trabajo de tesis "Una Taxonomía de procesos para la configuración de proyectos ágiles basados en Scrum" (Resolución 557/12).

Pero para el diseño de un prototipo de herramienta que automatice el proceso de configuración, es necesario resolver el punto 3. Nótese que este análisis deberá hacerse para cada herramienta ALM que se quiera considerar.

El presente trabajo se llevará a cabo utilizando RTC como herramienta ALM.

En primer lugar, se identificarán los componentes configurables que ofrece RTC, como punto de partida para analizar cualquier tipo de adaptación del proceso.

Luego, se mostrará un posible escenario en el cual, mediante una serie de preguntas y respuestas, un usuario con conocimientos del dominio del producto a desarrollar plantea la necesidad de adaptar el proceso ofrecido por la herramienta. Esa información se tomará como entrada para la herramienta generadora del proceso adaptado, la cual determinará cuáles son los cambios necesarios en RTC y generará el XML del proceso incluyendo dichos cambios.

Este XML resultante se incorporará a RTC, y se podrá ver cómo los cambios seleccionados fueron implementados.

Como punto de partida, la siguiente tabla describe cuáles son los elementos configurables del proceso provisto por RTC para un proceso Scrum estándar [IBM]. Cada fila representa los atributos que pueden editarse en base a las características del proyecto de desarrollo de software sobre el que se trabajará.

Table 1. Ítems configurables en RTC

Área	Categoría configurable	Ejemplo en RTC
Gestión de Proyecto	Estructura de iteraciones, planes y reportes, de acuerdo al tipo de proyecto: investigación; producto interno; producto final	- Desarrollo de producto: iteración de entrega e iteraciones de desarrollo de 2 a 4 semanas; uso de la métrica *sprint burndown*; - Investigación: iteraciones de entrega y desarrollo; gráfico de progreso de tareas por persona
	Versiones de producto (uno o más)	Línea de tiempo para tareas de mantenimiento
	Gestión de riesgos a nivel de producto	Generación de un nuevo tipo de ítem de trabajo: riesgo
Gestión de Producto	Documentación del producto	- Tareas de documentación como parte del sprint - Línea de Tiempo separada para equipos de documentación
	Gestión de requerimientos	Generación de un nuevo tipo de ítem de trabajo: requerimiento
	Creación de historias de usuario	- Consultas para validar completitud de las historias de usuario - Nuevas validaciones para campos requeridos
Gestión de versiones del producto	Gestión de Portfolio de producto	- Creación de una línea de tiempo separada por cada versión del producto - Creación de varias iteraciones, una por cada versión del producto en paralelo, en la misma línea de tiempo - Configuración de equipos para cada versión del producto
	Mantenimiento de versiones entregadas	- Línea de tiempo separada para mantenimiento - Configuración de campos para identificar la versión a la que corresponde cada ítem
	Verificación y Validación	- Línea de tiempo separada para tareas de verificación y validación

Área	Categoría configurable	Ejemplo en RTC
Equipos Scrum	Más de un equipo trabajando en el producto	- Jerarquía de equipos según funcionalidad o versión del producto
	Monitoreo de estado	Configuración de reportes estándares para monitoreo de progreso
	Equipo de Dueños de Producto y Scrum Masters	Creación de nuevos roles (Proxy de Dueño de Producto, Uber Scrum Master)

4. Resultados

Las opciones de configuración de RTC listadas en la sección anterior son la primera entrada necesaria para poder generar un proceso configurado en RTC de manera automática. Aún queda desarrollar la taxonomía en sí, que es la que va a recolectar la información de contexto necesaria para generar la configuración de procesos correcta para un proyecto dado.

Sin embargo, para poder llevar adelante la prueba de concepto, se puede asumir un escenario particular que involucre una sola pregunta y un sólo cambio de configuración en el proceso. Si este escenario puede automatizarse, entonces podemos asumir que siempre que la taxonomía pueda mapearse a una posible de configuración de RTC, entonces esa configuración se puede automatizar.

El escenario a demostrar es el siguiente: RTC ofrece un tipo de ítem de trabajo que representa los defectos del software reportados en un producto dado. Dicha entidad cuenta con los campos mostrados en la siguiente imagen[4].

[4] Las etiquetas de los campos están en inglés para simplificar el manejo de los archivos XML.

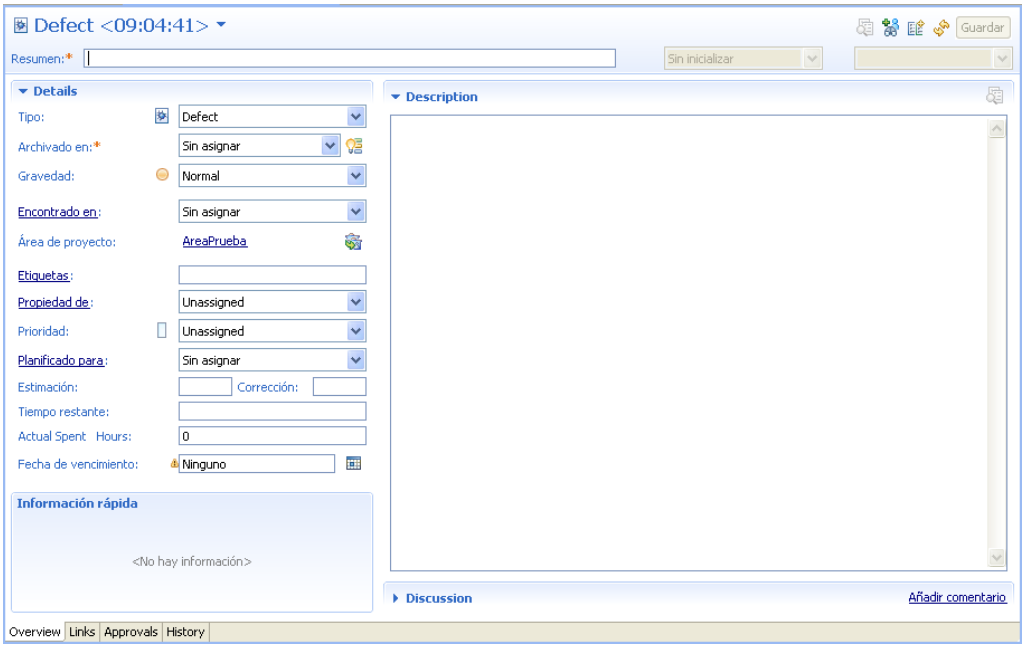


Fig. 1. Entidad de defectos original en RTC

Se pretende personalizar el proceso de manera tal que dicha entidad "defectos" incluya una nuevo campo que permita registrar el ID de dicho defecto en una herramienta externa. De esta manera, se pueden identificar así aquellos defectos encontrados internamente (donde el nuevo campo está vacío), de aquellos reportados por algún cliente externo (donde el campo guarda una referencia el mismo defecto en alguna otra herramienta).

Para el escenario de prueba, se parte de las siguientes preguntas al potencial usuario de la herramienta.

¿Gestiona los defectos encontrados por el cliente?

El objetivo de la pregunta es conocer si los defectos encontrados por el cliente se van a incluir como parte de trabajo del equipo que está desarrollando a su vez el producto. Así, se descartan las situaciones en las que los defectos son gestionados / corregidos en proyectos separados.

Fig. 2. Determinación del monitoreo de defectos del cliente

¿Los va a mantener en el mismo backlog del producto?

En caso de que la pregunta anterior fuera afirmativa, aún es necesario saber cómo se van a planificar las correcciones de defectos. Si se mantienen en el mismo backlog de producto, junto con las nuevas funcionalidades pendientes, se puede asumir que la corrección de defectos se planifica de la misma manera que el resto del trabajo a realizar para el producto, con los mismos tiempos planificados y similar proceso de desarrollo.

Fig. 3. Modo de gestión de los defectos reportados por el cliente

Ante la respuesta afirmativa a estas preguntas, lo que se determina es que los defectos encontrados y reportados por los clientes se gestionan como parte del ciclo normal de desarrollo del producto. En consecuencia, la herramienta agregará un nuevo campo de texto a la entidad de defectos para que, cada vez que se reporte en RTC un defecto que fue encontrado por el cliente, el mismo incluya a su vez el ID de dicho defecto en la herramienta que utiliza el cliente para reportar problemas.

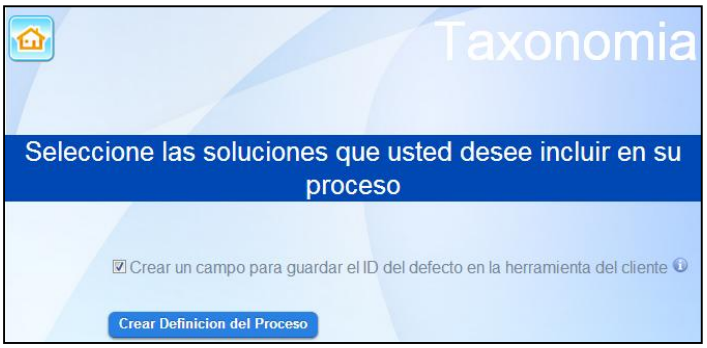


Fig. 4. Solución propuesta: Nuevo campo en la entidad de defectos

Ahora bien, como se puede apreciar en la **Fig. 4.** Solución propuesta: Nuevo campo en la entidad de defectos, antes de generar el XML con el proceso modificado de acuerdo a las respuestas dadas, la herramienta muestra al usuario cuáles son las soluciones propuestas.

Estas soluciones van a surgir como parte del proceso de generación de la taxonomía, a generarse como parte del trabajo de tesis de maestría "Una taxonomía de procesos para la configuración de proyectos ágiles basados en Scrum". Las mismas serán el resultado del análisis de las mejores prácticas utilizadas por proyectos Scrum para cada situación.

Sin embargo, al ser un compendio de buenas prácticas, la herramienta de configuración del proceso de RTC mostrará al usuario cuáles son los cambios recomendados a realizar en el proceso, pero será el usuario quien, en definitiva, podrá elegir qué cambios aplicar y qué cambios no.

Una vez confirmados los cambios, la herramienta generará el proceso completo que debe utilizarse para crear un proyecto en RTC, en formato XML.

Una vez que se aplica dicho proceso (representado en el XML) a un proyecto en RTC, la entidad de defectos para dicho proyecto incorpora un campo extra, denominado en este ejemplo *Customer Defect ID* (ID de Defectos del Cliente). Esto puede notarse en la siguiente imagen.

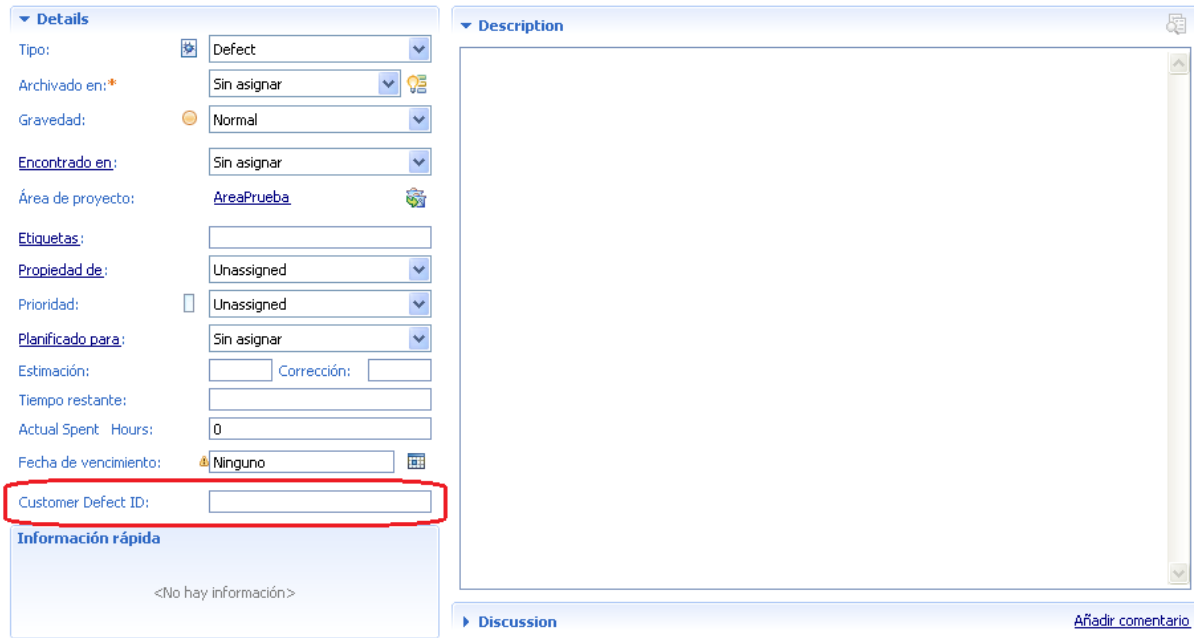


Fig. 5. Entidad de defectos resultante, luego de aplicar los cambios en el proceso de RTC

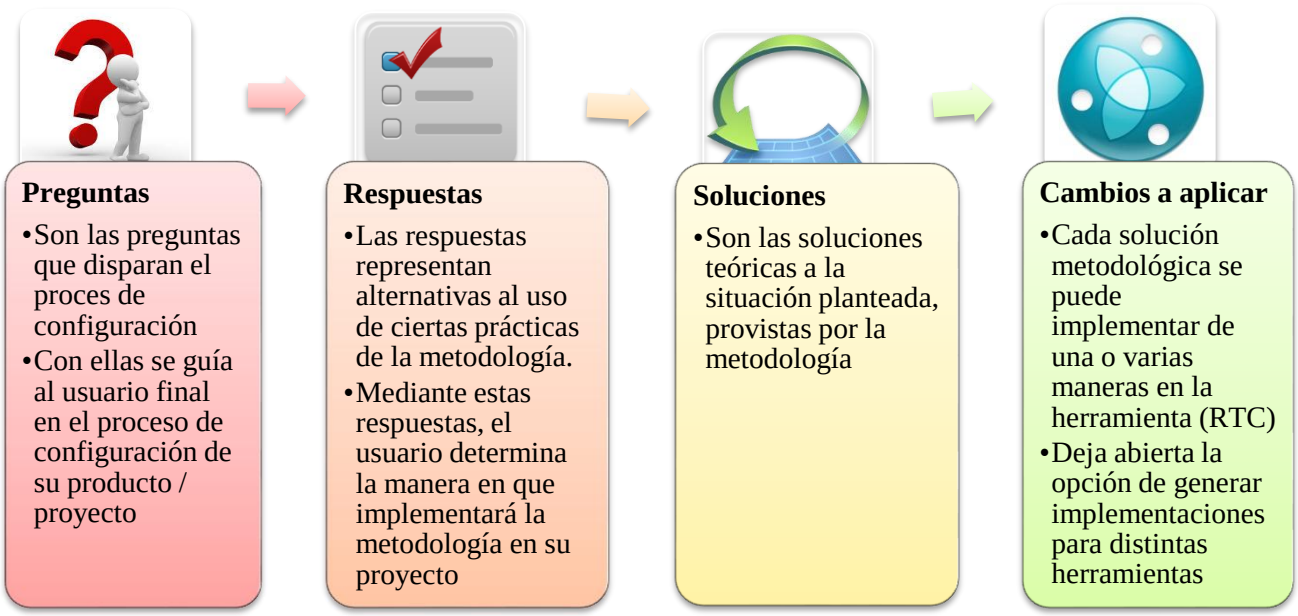
Configuración de la aplicación para la obtención de resultados

Para poder lograr los resultados mencionados en esta sección, es necesario configurar la herramienta de taxonomía, de manera de enlazar cada pregunta de la taxonomía con sus posibles soluciones, y los cambios específicos en el proceso que hacen falta para poder implementar dichas soluciones.

En esta versión de la herramienta, toda esta configuración se realiza a través de un archivo XML, en el que el potencial administrador de la herramienta instanciará la taxonomía.

Esta configuración provee una manera sencilla de mantener actualizada la taxonomía a medida que se obtenga más conocimientos sobre cierta área de la implementación de Scrum, o a medida que se sumen nuevas prácticas en su utilización.

El XML de configuración / instanciación de la taxonomía está formado por tres grandes bloques interrelacionados entre sí, tal como lo muestra la siguiente imagen.



En el XML, estas entidades se representan de la siguiente manera:

Preguntas:

```
<question id="1" question="¿Monitorea defectos encontrados por el cliente
en el backlog del producto?" isFirstQuestion="true">
      <answers>
            <answer>1</answer>
            <answer>2</answer>
      </answers>
</question>
```

- ***id***: identificador de cada pregunta. Sirve para referencias entre preguntas y para enlazar preguntas y respuestas.
- ***question***: texto de la pregunta que se muestra al usuario.
- ***isFirstQuestion***: atributo que sirve para señalar la pregunta con la que se inicia la taxonomía.
- ***answers***: conjunto de respuestas posibles para cada pregunta.

Respuestas

```
<answer id="1" answer="Si" nextQuestion="2">
            <solutions>
                  <solution>1</solution>
            </solutions>
      </answer>
```

- ***id:*** identificador de la respuesta. Sirve para referencias entre preguntas y para enlazar respuestas y soluciones.
- ***answer:*** texto de la respuesta que se muestra al usuario.
- ***nextQuestion:*** indica cuál es la siguiente pregunta en la taxonomía. De esta manera, de acuerdo a las respuestas seleccionadas, el conjunto final de preguntas utilizadas puede variar.
- ***solutions:*** conjunto de soluciones a aplicar para implementar la respuesta dada a la pregunta. Estas soluciones representan implementaciones particulares para una herramienta de gestión de proyectos en particular. En este trabajo, la herramienta utilizada es RTC, pero se podrían realizar implementaciones adicionales que contemplen otras herramientas.

Soluciones

```
<solution id="1" solution="Generar un campo para el registro del ID del
defecto en la herramienta del cliente">
            <changes>
                  <change>1</change>
            </changes>
      </solution>
```

- ***id***: identificador de la solución. Se utiliza para enlazar respuestas con sus respectivas soluciones.
- ***solution:*** texto descriptivo de lo que representa la solución. Esta descripción es agnóstica a la herramienta. Representa la solución teórica provista por la metodología.
- ***changes:*** cambios que hay que aplicar en el proceso para poder implementar esta solución. Estos cambios dependen de la herramienta de gestión de procesos que se esté utilizando.

Cambios a aplicar (RTC)

- ***id***: identificador del cambio. Sirve para asociar los cambios necesarios a cada solución propuesta.

- ***xPath***: RTC maneja el proceso en formato XML. Para poder aplicar cualquier tipo de cambio, es necesario conocer la porción del documento XML que se va a modificar. Dichas porciones pueden identificarse como una ruta interna, denominada XPATH. En este caso, el atributo determina el lugar exacto donde se va a aplicar el cambio.
- ***xmlData***: es la porción de XML a agregar en el proceso para implementar el cambio.

5. Conclusiones y trabajos futuros

El presente trabajo se realizó bajo la hipótesis de que es posible la configuración automática de una herramienta ALM dado el conocimiento de las características de un producto y un proyecto, y su relación con las entidades configurables de dicha herramienta.

Tal como se puede concluir a través de los resultados presentados, se ha logrado generar la configuración de una herramienta tomada como piloto (RTC para este trabajo) y automatizar su configuración basados en una taxonomía de conocimiento.

Asimismo, se trabajó en la realización de una aplicación genérica y que soporte distintos idiomas para la automatización de soluciones basadas en una taxonomía cualquiera sea el idioma o herramienta a configurar.

Actualmente el grupo se encuentra trabajando en el enriquecimiento de la taxonomía de conocimiento necesaria como entrada a la herramienta de automatización a través de encuestas y entrevistas en la industria local para luego realizar una prueba piloto en empresas del medio.

6. Bibliografía

(OMG), Object Management Group. 2005. *Meta Object Facility (MOF) 2.0 Query/View/Transformation/Specification.* s.l. : Object Management Group (OMG), 2005.

Agile Alliance. Agile Alliance. [Online] http://www.agilealliance.org/.

Agile Methods and CMMI: Compatibility or Conflict? **Fritzsche Martin, Keil Patrick. 2007.** s.l. : e-Informatica Software Engineering Journal, 2007.

Anis Ferchichi et al. 2007. *Implementing integration of quality standards CMMI and ISO 9001: 2000 for software engineering.* London, U.K. : Springer London, 2007. 978-1-84628-975-0.

Barry W. Boehm et al. 1984. *Software Development Environment for Improving Productivity.* California : IEEE Computer Society Press, 1984. Computer pp. 30-44.

Beck, Kent, et al. 2001. http://www.agilemanifesto.org/. [Online] Feb 13, 2001. [Cited: Abr 2, 2011.]

—. 2001. http://www.agilemanifesto.org/. [Online] Feb 13, 2001. [Cited: Abr 2, 2011.]

Black, Rex. 2009. *Advanced Software Testing vol.1.* Santa Barbara, USA : Rock Nook Inc., 2009. ISBN-13: 978-1-933952-19-2.

—. 2009. *Advanced Software Testing vol.2.* Santa Barbara, USA : Rock Nook Inc., 2009. ISBN-13: 978-1-933952-36-9.

Chameleon, Green. 2006. Green Chameleon. [Online] Abril 18, 2006. [Cited: Abril 14, 2012.] http://www.greenchameleon.com/gc/blog_detail/defining_taxonomy/.

CMMI Appraisal Program. Software Engineering Institute (SEI). Page 20. March 2008. *Process Maturity Profile CMMI® SCAMPISM Class A Appraisal Results 2007 Year-End Update.* Pittsburgh, Pennsylvania, USA : Carnegie Mellon University, Page 20. March 2008. http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2008MarCMMI.pdf.

CMMI Product Team. 2002. *Capability Maturity Model Integration Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1).* Pittsburgh, Pennsylvania, USA : Software Engineering Institute (SEI), 2002.

—. August 2006. *CMMI for Development, version 1.2.* Pittsburgh, Pennsylvania, USA : Software Engineering Institute (SEI), August 2006. CMU/SEI-2006-TR-008.

Curtis, W. May 1995. *Building a Cost-Benefit Case for Software Process Improvement.* Boston, MA : Notes from Tutorial given at the Seventh Software Engineering Process Group Conference, May 1995.

D. Rubio, N. Andriano, A. Ruiz de Mendarozqueta, C. Bartó. 2008. *An integrated improvement framework for sharing assessment lessons learned.* La Rioja : Proceedings del XIV Congreso Argentino de Ciencias de la Computación, 2008. ISBN 987-24611-0-2.

Donna K. Dunaway; Steve Masters. November 2001. *CMM®-Based Appraisal for Internal Process Improvement (CBA IPI) Version 1.2 Method Description.* Pittsburgh, USA : Software

Engineering Process Management Program, SEI, November 2001. CMU/SEI-2001-TR-033, ESC-TR-2001-033.

Dorothy Graham, Erik van Veenendaal, Isabel Evans and Rex Black. 2007. *Foundations of Software Testing ISTQB Certification.* London : Editorial: Thomson Learning, 2007. ISBN-13: 978-1-84480-355-2.

Glazer, Hillel, et al. 2008. *CMMI or Agile: Why Not Embrace Both!* s.l. : Carnegie Mellon University, 2008.

Haumer, Peter. 2007. Eclipse Process Framework Composer - Part 1Key Concepts. [Online] 04 2007. [Cited: 04 29, 2010.] http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf.

—. 2006. *Eclipse Process Framework Composer. Delta Version: From the IBM donation to Version 1.x.* 3 10, 2006.

Humphrey, Watts S. and Kellner, Marc I. 1989. *Software Process Modeling: Principles of Entity Process Models.* Pennsylvania : ACM, 1989. Proceedings of the 11th international conference on Software engineering, pp. 331-342.

Humphrey, Watts S. 1989. *CASE Planning and the Software Process.* s.l. : Software Engineering Institute - http://repository.cmu.edu/sei/96, 1989. Paper 96.

—. 1989. *Managing the Software Process.* Massachusetts : Addison-Wesley, 1989. ISBN: 0201180952.

IBM. Rational Team Concert. [Online] Rational Software. [Cited: 04 29, 2010.] http://www.ibm.com/developerworks/rational/products/rtc/.

IBM Rational Unified Process. IBM Rational Unified Process. [Online] http://www-01.ibm.com/software/awdtools/rup/.

IBM RTC, Jazz.net. 2009. Rational Team Concert. [Online] Jazz.net, Sep 2009. [Cited: Abr 3, 2011.] http://www-01.ibm.com/software/rational/products/rtc/.

—. 2011. Rational Team Concert Beta 3. [Online] Jazz.net, Mar 28, 2011. [Cited: Abr 3, 2011.] http://jazz.net/projects/rational-team-concert/.

IBM, Jazz. Help - IBM Rational Software. *IBM Rational Software.* [Online] [Cited: 05 04, 2013.] https://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/index.jsp.

—. Help - IBM Rational Software - Scrum Template. *IBM Rational Software.* [Online] IBM. [Cited: 05 04, 2013.] https://pic.dhe.ibm.com/infocenter/clmhelp/v4r0/topic/com.ibm.jazz.platform.doc/topics/r_scrum.html.

Institute of Electrical and Electronics Engineers, Inc. 2008. IEEE. [Online] IEEE, 2008. http://www.ieee.org/web/aboutus/home/index.html.

International Organization for Standardization. 2008. ISO. [Online] ISO, 2008. http://www.iso.org/iso/about.htm.

—. 2004. ISO 9126-4:2004, Software Engineering - Product Quality. 2004. ICS 35.080.

International Organization for Standarization. 2002. *ISO9001:2000 Sistemas de gestión de la calidad - Requisitos.* s.l. : ISO copyright office, 2002. ICS 01.040.03.

International Software Testing Qualifications Board. ISTQB - International Software Testing Qualifications Board. [Online] http://istqb.org/display/ISTQB/Home.

ISTQB. 2011. *Foundation Level Syllabus.* 2011.

J.L.Pfleeger. 2002. *Ingeniería del Software: Teoría y Prática.* Buenos Aires : Prentice Hall, 2002. ISBN: 8131720985.

Jacobson, James E. Dec., 1963. *The Wilcoxon Two-Sample Statistic: Tables and Bibliography.* s.l. : Journal of the American Statistical Association, Vol. 58, No. 304, pp. 1086-1103, Dec., 1963.

Jeannine M.Siviy; M. Lynn Penn; Robert Stoddard. 2007. *Achieving Success via Multi-Model Process Improvement.* Pittsburgh, Pennsylvania, USA : SEPG 2007. Carnegie Mellon University, 2007.

Jennifer Gremba and Chuch Myers. 1997. *The Ideal(SM) Model: A practical Guide for process improvement.* Pittsburgh, Pennsylvania, USA : Software Engineering Institute (SEI), Bridge, issue three, 1997.

Joao M. Fernandes - Mauro Almeida. 2010. *Classification and Comparison of Agile Methods.* Braga, Portugal : s.n., 2010.

Laycock, Martyn. 2005. *Collaborating to compete: achieving effective knowledge sharing in organizations.* The Learning Organization : Emerald Group Publishing Limited, 2005. DOI:10.1108/09696470510626739.

Linda Rising and Norman S. Janoff. 2000. *The Scrum Software Development Process for Small Teams.* s.l. : IEEE, 2000.

Lisa Crispin - Janet Gregory. 2009. *Agile Testing - A practical guide for testers and agile teams.* Boston : Pearson Education Inc., 2009. ISBN-13: 978-0-321-53446-0.

Mapping agile project management practices to project management - Challenges for software development. **Saya Poyu Sone. 2008.** Washington DC : s.n., 2008.

Marin, Mike. 2001. *The Workflow Management Coalition.* Florida : s.n., 2001.

Mark Paulk et all. February 1993. *Capability Maturity Model for Software, Version 1.1.* Pittsburgh, Pennsylvania, USA : Software Engineering Institute Carnegie Mellon University, February 1993. CMU/SEI-93-TR-24 ESC-TR-93-177.

Miller, Jerry. 2001. The Internet's Impact On Business Relationships. *Information Week.* [Online] Sears, Roebuck and Co, September 17, 2001. http://www.informationweek.com/news/management/showArticle.jhtml?articleID=6506627.

Mountain Goat Software, Imágenes. 2005. Mountain Goat Software - Imágenes. [Online] 2005. [Cited: Abr 2, 2011.] http://www.mountaingoatsoftware.com/system/asset/file/18/ScrumLargeNoLabels.png.

MSF Team, Microsoft. 2002. *MSF Process Model v. 3.1.* For more information on MSF, see: http://www.microsoft.com/msf : Microsoft, 2002.

O.J. Dahl, E. W. Dijkstra, C. A. R. Hoare. 1972. *Structured Programming.* London : Academic Press, 1972.

Object Management Group (OMG). 2007. *MOF 2.0/XMI Mapping.* s.l. : Object Management Group (OMG), 2007.

—. 2008. *Software and Systems Process Engineering. Meta-Model Specification.* s.l. : Object Management Group (OMG), 2008.

Oliveira, Sandro et al. 2010. *Mapeamento dos Conceitos do guia do CMMI em notações do SPEM no contexto da Definição do Processo de Software.* Lavras : InfoComp - Universidade Federal de Lavras, 2010. ISSN: 18074545.

OMG, Object Management Group. 2008. *Software & Systems Process Engineering Meta-Model Specification Version 2.0.* Standard document URL: http://www.omg.org/spec/SPEM/2.0/PDF : Object Management Group, 2008. OMG Document Number: formal/2008-04-0.

P. Szyrko, M. Silclir, G. García Favre, D. Rubio. 2009. *Un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software.* San Juan : Workshop de Investigadores en Ciencias de la Computación WICC09, 2009.

Peterson, David. 2009. What is Kanban? *Kanban Blog.* [Online] 2009. [Cited: May 01, 2013.] http://www.kanbanblog.com/explained/index.html.

Poppendiek, Mary and Poppendiek, Tom. 2003. *Lean Software Development: An Agile Toolkit.* New Jersey : Addison Wesley, 2003.

Pressman, Roger (5ª edición). 2002. *Ingeniería de Software.* Madrid : Mc Graw Hill, 2002.

Project Management Institute. PMI - Project Management Institute. [Online] http://www.pmi.org/.

—. 2004. *Project Management Body of Knowledge.* 2004.

Rico, David F. January 2004. *ROI of Software Process Improvement (Foreword by Roger S. Pressman).* s.l. : J. Ross Publishing, Inc., January 2004. ISBN: 1-932159-24-X.

Roger C. Schank. 2002. *Designing World-Class E-Learning: How IBM, GE, Harvard Business School and Columbia University Are Succeding at e-Learning.* s.l. : McGraw-Hill, 2002. ISBN:0-07-137772-7.

Rolland, C., Souveyet, C. and Moreno, M. 1995. *An Approach for Defining Ways-of-Working, Information Systems.* Oxford : Elsevier Science Ltd, 1995. Sixth International Conference on Advanced Information Systems Engineering, pp. 337-359.

Royce, Dr. Winston W. 1970. *Managing the Development of Large Software Systems.* pp. 1-9 : Proceedings of IEEE WESCON 26, 1970.

Ruiz, Francisco and Verdugo, Javier. 2008. *Guía de Uso de SPEM 2 con EPF Composer.* s.l. : Universidad de Castilla-La Mancha. Escuela Superior de Informática. Departamento de Tecnologías y Sistemas de Información. Grupo Alarcos, 2008.

Saul Carliner, Patti Shank. April 2008. *The E-Learning Handbook: A Comprehensive Guide to Online Learning (Hardcover).* s.l. : Pfeiffer, April 2008. ISBN-10: 0787978310.

Scacchi, Walt. 2002. *Process Models in Software Engineering.* Wiley : Encyclopedia of Software Engineering, 2nd. Edition, 2002. 993-1005.

SCAMPI Upgrade Team. August 2006. *Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2: Method Definition Document.* Pittsburgh, Pennsylvania, USA : Software Engineering Institute, August 2006. HANDBOOK CMU/SEI-2006-HB-002.

Schwaber, Ken. 2004. *Agile Project Management with Scrum.* s.l. : Microsoft Press, 2004. ISBN 0-7356-1993-X.

Schwaber, Ken and Beedle, Mike. 2002. *Agile Software Development with Scrum.* Upper Saddle River, NJ : Prentice Hall, 2002. 0130676349.

Schwaber, Ken. 2001. Scrum Practices. [book auth.] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum.* Upper Saddle River, NJ : Prentice Hall, 2001.

Seung-Hun, Park, et al. 2008. Applying UML and software simulation for process definition, verification, and validation. s.l. : KAIST SE LAB, 8 6, 2008.

Silclir, Mauricio, et al. 2010. *Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software.* Ciudad Autónoma de Buenos Aires : ASE 2010, 2010.

Software Engineering Institute (SEI). 2008. [Online] Carnegie Mellon University, 2008. http://www.sei.cmu.edu/.

Software Tech News. March 2007. *Performance Outcomes from Process Improvement.* s.l. : The Data and Analysis Center for Software (DACS), March 2007.

Software Technology Support Center. January 2002. *Mappings of CMMI-SE/SW Version 1.1 and SW-CMM Version 1.1.* s.l. : http://www.sei.cmu.edu/cmmi/adoption/pdf/stsc-mappings.pdf, January 2002.

Sommerville, Ian. 2007. *Ingeniería de Software (7ª edición).* Madrid : Addison Wesley, 2007.

—. 2010. *Software Engineering (9th Edition).* Madrid : Addison Wesley, 2010. ISBN: 0137035152.

Stenning, V. 1987. *On the role of an environment.* California : IEEE Computer Society Press, 1987. Proceedings of the 9th international conference on Software Engineering, pp. 30-35.

Tague, Nancy N. 2004. *The Quality Toolbox, Second Edition.* s.l. : ASQ Quality Press, 2004. ISBN 9780873896399, pp. 219-223.

Takeuchi, Hirotaka and Nonaka, Ikujiro. 1986. *The New New Development Game.* s.l. : Harvard Business Review, 1986.

2010. Team Center para Team Foundation Server. *MSDN.* [Online] Microsoft Corporation, 2010. [Cited: 04 29, 2010.] http://www.microsoft.com/spanish/msdn/latam/vstudio/teamsystem/team/.

Thomas McGibbon; Daniel Ferens; Robert L. Vienneau. 2007. *A Business Case for Software Process Improvement (2007 Update).* s.l. : Measuring Return on Investment from Software Engineering and Management, 2007. DACS Report Number 347616.

Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software. **Silclir, Mauricio, et al. 2010.** Córdoba : s.n., 2010. JAIIO 2010. p. 14.

Universidad Tecnológica Nacional-Facultad Regional Córdoba. [Online] [Cited: 10 9, 2009.] http://www.frc.utn.edu.ar/.

Well, Don. 2009. Extreme Programming: A gentle introduction. *Extreme Programming.* [Online] Sep 28, 2009. [Cited: May 01, 2013.] http://www.extremeprogramming.org/.

Wikipedia, (Español). 2001. http://es.wikipedia.org/wiki/Rugby. [Online] Wikipedia.org, Septiembre 28, 2001. [Cited: Abril 2, 2011.]