Universidad Tecnológica Nacional

Facultad Regional Córdoba

Dirección de Posgrado

Especialización en Ingeniería en Sistemas de Información

Selección de métodos y herramientas de simulación y definición de procesos de desarrollo de software para generación automática de entrenamiento

Claudio Javier González

D.N.I.:30660260

Email: claudiojgonzalez@gmail.com

Córdoba, Febrero de 2013

Índice

Índice de Figuras

Índice de Tablas

Resumen

El presente trabajo tiene como objetivo la selección de las herramientas necesarias para la generación simulaciones que soporten el entrenamiento en procesos de desarrollo de software. Se analizaran diferentes herramientas de definición de procesos de software, basados en SPEM, y de simulación para luego aplicar un método de decisión formal para lograr la selección.

Palabras claves: Entrenamiento Electrónico, Simulación, Procesos de desarrollo de software, SPEM.

Abstract

The present work aims at the selection of the necessary tools to generate simulations that support training in software development processes. We will analyze different tools software process definition, based on SPEM, simulation and then applying a formal decision method to get the selection.

Keywords: e-learning, simulation, software development process, SPEM.

1. Introducción

El presente trabajo constituye la primera etapa del plan de tesis de maestría en Ingeniería en Sistemas de Información, "**Propuesta de un modelo de generación automática de e-learning de simulación para Ingeniería de Software"**. El mismo tiene por objetivo la creación de un modelo que soporte la generación automática de un entrenamiento electrónico (de ahora en más denominado e-learning) de simulación de un proyecto de software a partir de la definición de un proceso. Dado que existen diferentes tecnologías de simulación y de herramientas de definición de procesos el primer objetivo del plan de tesis de maestría es realizar una revisión bibliográfica de un subconjunto de las opciones disponibles y, luego, la selección de la herramienta de definición de proceso y la tecnología de simulación.

En base a lo mencionado anteriormente se plantea como objetivo del presente trabajo realizar una exploración bibliográfica con el fin de lograr la selección de los métodos y herramientas de simulación y definición de procesos de desarrollo de software que son necesarios para la realización de dicha herramienta. El trabajo será elaborado en diferentes secciones, una primera sección que introducirá a e-learning, simulación y procesos de software. Luego se presentaran el método y los criterios aplicados para la selección de las herramientas, acompañados de la selección. Una tercera sección donde se describirán las herramientas que serán objeto de análisis. Por último, se desarrollarán las conclusiones del trabajo y detallarán los pasos a seguir en las próximas etapas.

2. Introducción a E-Learning y Simulación

a. E-Learning

Desde su aparición, el concepto de "E-Learning" se asoció a diferentes metodologías que aprovechaban la tecnología de información con fines educativos, así, el diccionario de Oxford define E-Learning como "Enseñanza realizada a través de medios electrónicos, típicamente en Internet". Por ello, y debido al rápido avance de la tecnología, se dio carácter de e-learning a métodos que si bien utilizan la tecnología de la información, tienen objetivos, herramientas y fundamentos pedagógicos distantes. En el trabajo de Roderick Munro (Munro, 2005), podemos apreciar que

dentro del universo de los e-learnings encontramos desde simples presentaciones a través de la web hasta complejos simuladores de vuelo.

De todos modos, el aspecto que comparten los métodos de e-learning es que son aplicados para lograr que el estudiante aprenda en forma total o parcialmente desatendida utilizando las tecnologías informáticas para lograr la distribución de la información, la construcción del ambiente de aprendizaje y la posterior evaluación. Lo cual marca una diferencia clave con la técnica clásica de enseñanza donde el rol del Profesor forma una parte clave (NTL Institute, 2012) (Laycock, 2005).

En primera instancia, la metodología de e-learning fue ampliamente usada con las formas clásicas Profesor-Alumno, por ejemplo, en enseñanza a distancia donde alumnos en diferentes ubicaciones toman clases en un salón virtual creado en la web. Este fue uno de los primeros usos por parte de compañías globales y universidades que aprovecharon la web para poder distribuir los cursos de manera más efectiva (Henderson, 2003).

Luego, con el avance de las tecnologías de información, aparecieron otros tipos de e-learning que eran fundamentados en metodologías de enseñanza diferentes. Así, aparecieron e-learnings que lograban simular el ambiente real, donde el estudiante podía ejercitar sus conocimientos pudiendo fallar sin consecuencias reales, y logrando así aprender tanto de los aciertos como de los fallos (Lin, Wen-Hsiang, Zhi-Wei, & Don-Lin, 2008) (Shank, 2001). En este tipo de simulaciones no existe un único camino para llegar al resultado final sino que cada estudiante puede resolver el problema de maneras diferentes y la simulación se adapta para llegar a un resultado final que puede ser satisfactorio o erróneo.

En el contexto de nuestro trabajo entendemos por "E-Learning" un programa de computadoras que enseña al alumno usando una simulación interactiva de una situación real, como es sugerido por Ruth Thomas (Thomas, 2001).

"…entendemos por "E-Learning" un programa de computadoras que enseña al alumno usando una simulación interactiva de una situación real…"

b. Simulación

Comenzaremos definiendo simulación tomando las palabras de R.E. Shannon (Shannon, 1975):

"La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema".

Sobre esta definición conceptual, podemos ver las simulaciones en tres categorías: Simulaciones Vivas, Simulaciones Virtuales y Simulaciones Constructivas. Las diferencias entre ellas se dan por el grado de participación humana y de equipamiento. Las primeras requieren alta participación de humanos y equipamiento real, por ejemplo juegos de guerra en entrenamientos militares. Las segundas son simulaciones en donde una persona o un conjunto de personas interactúan en un mundo virtual. Las últimas son simulaciones de modelos climáticos que dado un conjunto de valores de las variables iniciales arrojan un resultado final, y donde la interacción con personas es inexistente. En estos grupos podemos encontrar ejemplos de simulaciones usadas con diferentes fines, como investigación, desarrollo de nuevas tecnologías, entretenimiento y educación o entrenamiento (DoD, 1998) (University of Central Columbia).

A los fines de este trabajo se toma como eje las simulaciones virtuales utilizadas con fines educativos o de entrenamiento (González, Izaurralde, Marzo, & Rubio, 2009). Las simulaciones en entornos de entrenamiento, son comúnmente usadas cuando es difícil, costoso o peligroso permitir a los participantes del entrenamiento interactuar con el sistema real. Uno de los ejemplos más claros de simulaciones en entornos de entrenamiento son los simuladores de vuelo y las simulaciones de ejercicios militares (DoD, 1998). Las simulaciones proveen una representación interactiva de la realidad que permite a los estudiantes probar y descubrir cómo funciona o cómo se comporta un fenómeno, qué lo afecta y qué impacto tiene sobre otros fenómenos. El uso de este tipo de herramienta educativa alienta al estudiante para que manipule un modelo de la realidad y logre la comprensión de los efectos de su manipulación mediante un proceso de ensayo-error (Shank, 2001).

"…La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema…"

c. ¿Por qué incluir simulaciones en el entrenamiento?

Las investigaciones efectuadas por Nick van Dam (Van Dam, 2004) acerca de las tasas de retención, memoria y aprendizaje de las personas (Figura 1), indican que el ejercicio de una actividad es fundamental para garantizar la mayor retención posible durante el aprendizaje. No solo basta con tomar clases complementadas con material de soporte si no que el ejercicio de lo aprendido es de vital importancia para lograr una mayor efectividad del entrenamiento.

Estos estudios se complementan con los trabajos de NTL Institute (NTL Institute, 2012), sostienen que durante una sesión de aprendizaje se retiene el 5% de lo que oímos, el 10% de lo que leemos, el 75% de lo que practicamos y el 80% de lo que enseñamos a otra persona. Incluyendo a la escala propuesta por Nick Van Dam el aspecto social de interacción para enseñar a otra persona (ver figura 1).

"…permite sumergir al alumno en un ambiente de situaciones controladas, una simulación interactiva, sin ningún tipo de castigo ni reprobaciones, sólo mostrándole las consecuencias de sus errores y cómo prevenirlos…"

Por otra parte, la metodología de "Learning by Doing" (Shank, 2001) (aprender haciendo), recomienda sumergir al alumno en un ambiente de situaciones controladas, una simulación interactiva, sin ningún tipo de castigo ni reprobaciones, sólo mostrándole las consecuencias de sus errores y cómo prevenirlos, dando la libertad al alumno para que pueda aprender tanto de sus aciertos y en especial de sus errores. Esta metodología resalta la importancia de aprender de excepciones o errores, ya que el impacto emocional que estos provocan en el alumno es mayor que si todo ocurriese

siguiendo un curso normal. Es importante destacar que contar con un ambiente simulado es de vital importancia para poder aplicar esta metodología, ya que dejar que un alumno falle en la realidad podría tener consecuencias indeseadas.

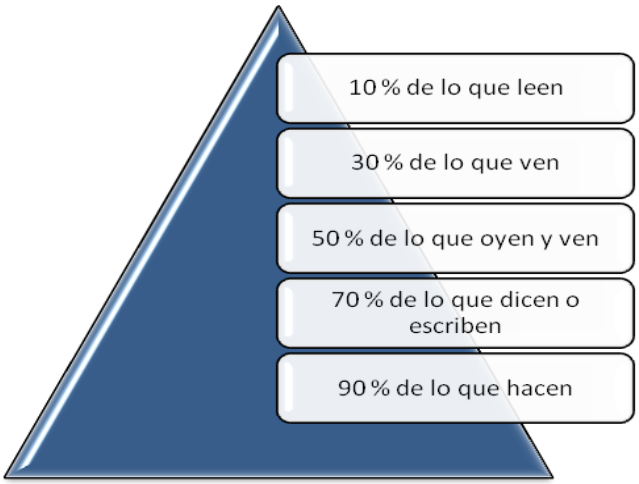

Figura 1: Pirámide de retención de aprendizaje

Estos autores nombrados con anterioridad se enfocan en la importancia de la práctica sin riesgos y poder compartir, ayudar o enseñar a otro para maximizar la efectividad del entrenamiento.
Hasta este momento se mencionó que el trabajo de tesis se centra en simulaciones interactivas con fines de educación o entrenamiento. El espectro de entrenamiento es amplio y no es la intención de este trabajo cubrirlo en su totalidad. Por ello queremos limitar el entrenamiento a entrenamiento organizacional (CMMI Product Team, 2006), que queda definido como el entrenamiento que una organización provee a sus trabajadores para realizar el trabajo según los procesos y cultura empresariales.

d. Efectividad de simulaciones interactivas en entornos educativos

Se comenzará tomando como referencia los trabajos de Kurt Squire (Squire, Barnett, M. Grant, & Higginbotham, 2004). En estos trabajos se usó una simulación llamada “Supercharged”, desarrollada en MIT (MIT, 2012) para enseñar sobre fuerzas electromagnéticas. Su objetivo era probar la efectividad de las simulaciones y para ello uso dos grupos de estudiantes, en uno de los grupos usó la simulación y en el otro el enfoque tradicional (Aldrich, 2009). Usando pre y post evaluaciones en los grupos de control, se encontró que los participantes que recibieron clases

tradicionales mejoraron su entendimiento en un 15% sobre su pre-evaluación, mientras que los grupos que usaron la simulación tuvieron un incremento del 28%.

En otro caso el Dr. John Dunning, profesor en Troy University (Troy University, 2012), descubrió que los estudiantes en una clase de comportamiento de organizaciones públicas que usaba métodos tradicionales tenían altas calificaciones. Sin embargo, cuando él encuestaba a los estudiantes seis meses después de que el curso había finalizado, los conocimientos y teorías enseñados no eran aplicados en el trabajo. Para probar el uso de simuladores el Dr. Dunning formó dos grupos, uno usaba la curricula tradicional y el otro guidado por simuladores. Seis meses después que el curso había finalizado, él encuesto a los estudiantes nuevamente y las diferencias entre los grupos fueron significativas. El grupo que siguió la metodología tradicional tuvo resultados similares que los grupos anteriores, pero el grupo que usó simuladores pudo demostrar la aplicación de los conocimientos en el ambiente de trabajo además de poder explicar los conceptos teóricos (Aldrich, 2009).

Ejemplos como estos se encuentran también en escuelas de negocio y organizaciones de ingeniería (Roberto, 2009), donde los ingenieros y/o estudiantes se sumergen en una simulación previamente a participar de proyectos reales con el objetivo de enseñar aspectos clave del trabajo a realizar.

En el área de la ingeniería de software, el trabajo “Experiencia de la Aplicación de Aprendizaje Activo en un Marco Universidad-Empresa” (González, Izaurralde, Marzo, & Rubio, 2009), presenta la simulación de un proyecto de software acotada a las actividades definidas en el área de procesos “Planificación de Proyectos” de CMMI (CMMI Product Team, 2006). Las conclusiones arrojadas por el trabajo determinaron que la metodología de “Learning by Doing” implementada como una simulación, fue efectiva para lograr el entendimiento y aprendizaje de una práctica. Además, se considera que la experiencia lograda fue positiva, ya que la aplicación motivó a estudiantes y profesionales a llegar a un final exitoso, produciendo por medio de la práctica, un incremento en los conocimientos del tema.

3. Introducción a procesos de desarrollo de software

a. Procesos de Desarrollo de Software

Un proceso de desarrollo de software es el marco de trabajo establecido, técnico y de gestión, para aplicar herramientas, métodos y personas a las actividades de software. La mayoría de los proyectos de software involucra el trabajo de varias personas. Cuando este grupo es relativamente pequeño, la coordinación y cooperación de las partes puede llevarse adelante de alguna manera más o menos informal. Pero cuando los proyectos son más grandes e involucran un número mayor de integrantes y/o de tareas más complejas y sofisticadas, la colaboración y coordinación dejan de ser tareas triviales y se hace evidente la necesidad de contar con algún mecanismo más formal: un proceso definido de desarrollo de software (Humphrey, Managing the software process, 1989).

"…UN proceso de desarrollo de software es el marco de trabajo establecido, técnico y de gestión, para aplicar herramientas, métodos y personas a las actividades de software …"

Dado que para el cumplimiento del objetivo de la tesis de maestría es necesario definir una herramienta de definición de procesos, en la presente sección se introducirá conceptualmente el modelo SPEM (Meta-Modelo de Procesos de Ingeniería de Software y Sistemas) que se usa como base para las herramientas de definición de procesos de desarrollo de software que serán analizadas más adelante. SPEM tiene como objetivo específico la definición de procesos de desarrollo de software y por ello logra representar con mayor fidelidad los estos tipos particulares de proceso de negocio (Portela, y otros, 2012).

b. Meta-Modelo de Procesos de Ingeniería de Software y Sistemas

El meta-modelo de Procesos de Ingeniería de Software y Sistemas (SPEM por sus siglas en idioma inglés) (OMG, 2008) es un marco conceptual que puede proporcionar los conceptos necesarios para modelar, documentar, presentar, gestionar, intercambiar, y la promulgar los métodos y procesos de desarrollo.

"…es un marco conceptual que puede proporcionar los conceptos necesarios para modelar, documentar, presentar, gestionar, intercambiar, y la promulgar los métodos y procesos de desarrollo…"

Un Meta-modelo describe un conjunto de conceptos genéricos y sus interrelaciones, que sirven de base para la definición de Modelos de un cierto Dominio. Por tanto, un metamodelo es un modelo de modelos. SPEM permite estandarizar la representación y gestionar bibliotecas de contenido reutilizable que luego puede ser usado en la definición de diferentes procesos de desarrollo según las necesidades particulares de los proyectos de software. En la siguiente figura podemos apreciar los elementos que componen el meta-modelo.

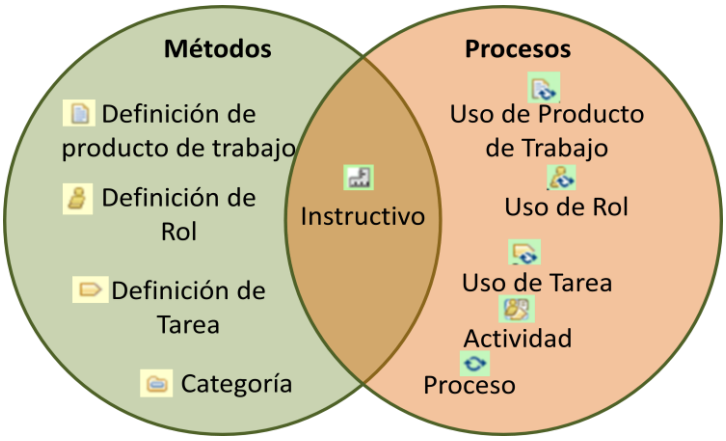

Figura 2: Elementos definidos por SPEM

La especificación del meta-modelo agrupa estos elementos en un conjunto de paquetes (OMG, 2010) que son descriptos a continuación.

i. Core

El paquete Core contiene las clases y las abstracciones (OMG, 2010) que forman la base para las clases en todos los demás paquetes del meta-modelo. Core define principalmente dos clases de capacidades SPEM 2.0: (1) La capacidad para un usuario de SPEM 2.0 para crear cualificaciones para las clases SPEM 2.0 permitiendo a los usuarios distinguir diferentes "Tipos" (OMG, 2010) de

instancias de clase SPEM 2.0. (2) Un conjunto de clases abstractas para definir los procesos expresados como SPEM 2.0.

ii. Estructura del proceso

Este paquete define la base de todos los modelos de procesos. Apunta a la creación de modelos de procesos simples y flexibles. Su estructura de datos básicos es una descomposición de las actividades anidadas que mantienen listas de referencias a la realización de clases de funciones, así como la entrada y salida de productos para cada actividad. Además que, proporciona mecanismos para la reutilización de procesos. Estas estructuras se utilizan para la representación de alto nivel de procesos que no están documentados textualmente.

iii. Comportamiento de proceso

Los conceptos del paquete "Estructura del Proceso" representan un proceso como una estructura de división estática, permitiendo la anidación de las actividades y la definición de las dependencias de precedencia entre ellas. El paquete de comportamiento de los procesos permite extender estas estructuras con los modelos de comportamiento (OMG, 2010). Sin embargo, no define el modo de modelado, sino que ofrece "links" a modelos de comportamiento externamente definidas, lo que permite la reutilización otras especificaciones ya sean estas OMG o de terceros. Por ejemplo, un proceso definido con los conceptos del paquete "estructura del proceso" puede estar vinculado a diagramas de actividad UML 2 (OMG, 2010) que representan el comportamiento de dicho proceso, o un producto de trabajo "Definición" del paquete de "Especificación de los Métodos" puede estar vinculado a un modelo de máquina de estados que representa el ciclo de vida del concepto modelado.

iv. Contenido Administrado

Los procesos de desarrollo no son, en muchos casos, sólo representados como modelos, si no que son documentados y administrados como descripciones en lenguaje natural. Para muchos métodos de desarrollo de software, es más importante documentación comprensible por humanos (que provea guías explicativas de las buenas prácticas de desarrollo de software) que modelos precisos. Las razones para esto son que muchos enfoques de desarrollo ven el desarrollo de software como un proceso creativo que requiere una constante re-evaluación en lugar de una secuencia estricta

de las actividades. El paquete de "contenido administrado" introduce conceptos para gestionar el contenido textual de tales descripciones. Estos conceptos se pueden utilizar ya sea independiente o en combinación con los conceptos del paquete de "estructura de proceso". Por ejemplo, un proceso basado en SPEM 2.0 podría estar compuesto únicamente por un conjunto de instancias de la meta-clase de "guía" definiendo las mejores prácticas de desarrollo en formato textual.

v. Contenido Método

Proporciona los conceptos para que los usuarios SPEM 2.0 puedan construir una base de conocimiento sobre el desarrollo, independiente de cualquier proceso y/o proyecto específico. Añade conceptos para definir la vida útil y los métodos de elementos de contenido, reutilizables en procesos, que proporcionan una base de conocimientos documentados de los métodos de desarrollo de software, las técnicas y las realizaciones concretas de mejores prácticas. Método de contenido se compone de explicaciones paso a paso que describen como objetivos específicos son logrados, independientemente de cómo son usados en un ciclo de vida de desarrollo específico. Los procesos reúsan estos elementos y los relacionan en secuencias parcialmente ordenadas que se adaptan a tipos específicos de proyectos.

vi. Proceso Con Métodos

Aquí se redefinen los procesos definidos en el paquete "Estructura de Proceso" con elementos del paquete "Contenido de Métodos". Considerando que el paquete "Contenido de Métodos" define los métodos y técnicas fundamentales para desarrollo de software, los procesos colocan estos métodos y técnicas en el contexto de un modelo de ciclo de vida que comprende, por ejemplo, las fases e hitos. Al aplicar los elementos del paquete "Contenido de Métodos", como las tareas, funciones y productos de trabajo a partes específicas del proceso, se crean clases de referencia (denominadas "Uso de Contenido de Métodos") que pueden almacenar información de cómo y qué partes del método se aplicará en ese punto en particular en el proceso.

vii. Plug-in de Métodos

Este paquete introduce conceptos para diseñar y administrar librerías de métodos y procesos fáciles de mantener, a gran escala, reutilizables y configurables. Los conceptos introducidos en este

paquete permiten la organización de las diferentes partes de dicha biblioteca en diferentes capas, en forma similar a las arquitecturas de software en capas. Los elementos de este paquete nos permiten seleccionar un subconjunto de las capacidades de los métodos y procesos llamados configuraciones de método. Sólo las capacidades seleccionadas serán visibles para el usuario final, permitiendo a los autores de proceso definir procesos consistentes y fáciles de mantener orientados a diferentes públicos pero que se pueden configurar para satisfacer las necesidades específicas del usuario final.

4. Selección de herramienta de definición de procesos

Como se anticipó en la introducción del trabajo, el objetivo final es la selección de un conjunto de herramientas que soporten la generación de una simulación de un proceso de desarrollo de software. Como primera medida se trabajó en la selección de una herramienta de definición de procesos que nos permitirá luego automatizar la generación de simulaciones, a continuación se explicará el método a ser aplicado para la selección de la herramienta.

a. Método para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software

Para llevar a cabo la selección de la herramienta de definición de procesos se usara el método publicado en "Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software" (Mauricio Silclir, 2010). Este trabajo define un método para la selección de herramientas de definición de procesos y los requerimientos que estas deben cumplir.

"…Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software define un método para la selección de herramientas de definición de procesos y los requerimientos que estas deben cumplir…"

Así, el modelo planteado consta de tres etapas que se explican a continuación:

1 - Definición de requerimientos y categorización: El análisis de las herramientas de modelado es planteado sobre la comparación de un conjunto de requerimientos considerados como

deseables para ser cumplimentados por las herramientas de modelado de proceso bajo análisis. Cada uno de estos requerimientos es categorizado con el fin de establecer una jerarquía entre aquellos requerimientos de alto valor contra aquellos que representan aspectos deseables pero cuyo cumplimiento es ciertamente opcional.

Un punto importante a considerar es que la definición de las categorías se realizó en base a dos fuentes principales. La primera de ellas fueron las necesidades particulares establecidas en torno a los objetivos del proyecto de investigación a partir del cual se genera el trabajo, los cuales pueden ser consultados en (P. Szyrko, Un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software., 2009). La segunda fuente considerada fueron las necesidades de la industria local, de acuerdo a los resultados de la investigación citada en (D. Rubio, 2008). Básicamente el objetivo del proyecto citado en (P. Szyrko, Un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software, 2009) plantea la validación automática de un proceso de desarrollo de software, y en consecuencia surge la necesidad de interpretar automáticamente el proceso definido. En este punto se encuentra una coincidencia importante con respecto del proyecto de generación de entrenamientos en el cual se desarrolla la especialidad. Ambos proyectos tienen como entrada principal una definición de procesos que debe ser interpretada, esta coincidencia justifica que los criterios definidos en (Mauricio Silclir, 2010) apliquen también al presente trabajo.

Adicionalmente, a cada categoría se le asigna un valor cuantitativo que será utilizado posteriormente al ejecutar el método cuantitativo de selección.

Las categorías en las que se clasifica cada uno de los requerimientos son:

- Mandatorio: indica un requerimiento que necesariamente debe ser cumplimentado por la herramienta. El valor cuantitativo asignado es 5.
- Alto valor: indica un requerimiento que en caso de ser cumplimentado por la herramienta que proporciona gran valor al proceso de modelado, sin que ello implique que sea mandatorio. El valor cuantitativo asignado es 3.
- Bajo valor: indica un requerimiento deseable en la herramienta, pero con un factor de incidencia reducido en el proceso de análisis y comparación. El valor cuantitativo asignado es 1.

2 - Análisis individual de cada herramienta: Una vez establecidos los requerimientos se analiza cada una de las herramientas bajo análisis valorando el cumplimiento de cada uno de esos requerimientos contra su implementación particular. Cada una de estas valoraciones tiene asociado un valor cuantitativo que será utilizado al aplicar el método cuantitativo de selección:

- Cumplimiento absoluto: la herramienta cumple con todas las especificaciones del requerimiento de forma directa. El valor cuantitativo asignado es 5.
- Cumplimiento parcial: la herramienta cumple con las especificaciones del requerimiento ya sea en forma parcial, considerando sólo algunas condiciones o aspectos de dicho requerimiento, o a través de la incorporación de algún plugin o siguiendo un procedimiento alternativo, comúnmente denominado "workaround". El valor cuantitativo asignado es 2.
- No cumplimiento: la herramienta no proporciona ningún nivel de cumplimiento del requerimiento. El valor cuantitativo asignado es 0.

3 - Análisis comparativo de herramientas: En esta etapa se resumen los resultados obtenidos del análisis de cada una de las herramientas y se procede a aplicar las operaciones numéricas tendientes a seleccionar una de ellas, obteniendo una valoración general de cada una de las herramientas sobre la base de estas dos dimensiones:

- Valor de cada requerimiento
- Cumplimiento del requerimiento en la implementación particular (herramienta)

Los pasos del método cuantitativo de selección son:

- Valor de cada requerimiento

Si una herramienta para un requerimiento particular categorizado como mandatorio ha sido valorado con un No Cumplimiento, queda automáticamente descartada.

- Cumplimiento del requerimiento en la implementación particular (herramienta)

Se obtiene el valor cuantitativo de análisis de cada una de las herramientas que han pasado la etapa aplicando la siguiente fórmula:

CRi: Categoría asignada al requerimiento i

VRij: Valoración aplicada para el requerimiento i en la herramienta j

Vj: Valoración cuantitativa de la herramienta

Vj= i=1..n ∑ CRi *VRij

- Selección de herramienta

Finalmente se comparan los valores cuantitativos de cada una de las herramientas y se selecciona la que mayor valor tenga.

b. Criterios de evaluación para la selección de la herramienta

A continuación se detallan los requerimientos específicos, definidos en (Mauricio Silclir, 2010), a tener en cuenta al momento de analizar y comparar las diferentes herramientas de modelado de procesos de desarrollo de software y la categoría correspondiente asignada. Esto corresponde al primer paso de la metodología planteada previamente.

- **La herramienta debe permitir definir nuevos procesos de desarrollo – Mandatorio**

Esto es una característica básica que debe ser cumplimentada por la herramienta, proporcionando la capacidad de definir procesos de desarrollo de software "desde cero", sin que ello implique la necesidad de realizar configuraciones y establecer dependencias con componentes externos al momento de proceder a la definición de dichos procesos.

- **La herramienta debe permitir adaptar procesos de desarrollo definidos – Mandatorio**

En este punto estamos asumiendo la existencia de procesos de desarrollo previamente definidos a los cuales se pretende incorporar cambios: ya sea eliminar ciertas partes que no se consideran aplicables, incorporar nuevas, o efectuar modificaciones sobre partes ya definidas. Este

requerimiento caracterizará en gran medida la flexibilidad y extensibilidad que proporciona la herramienta.

- **La herramienta debe permitir generar e incorporar componentes de proceso reusables – Alto valor**

Este punto caracteriza la reusabilidad que permite la herramienta. La premisa inicial es que ciertos componentes de proceso ya definidos estén disponibles para ser utilizados de forma transparente en la definición de otros procesos. Esto puede considerarse desde dos perspectivas diferentes: por una lado la posibilidad que brinda la herramienta de definir componentes genéricos aislados de cualquier definición de proceso y cuya finalidad es la de ser importados y utilizados al momento de proceder a la efectiva definición de un proceso de desarrollo de software. Por el otro, la posibilidad de extraer componentes de un proceso de desarrollo ya definido para incorporarlo en otro, representando otro nivel de reusabilidad.

- **La herramienta debe ser capaz de separar la definición de procesos de desarrollo de software de sus implementaciones particulares – Alto valor**

Una organización que desarrolla software define un proceso o un conjunto de procesos de desarrollo de software estándares, los cuales serán implementados en forma particular en cada uno de los proyectos que se lleven a cabo. La herramienta debe permitir por un lado la especificación de los procesos de software base dentro de la organización y por el otro la definición de cómo cada proyecto implementa dichos procesos. De esta forma la herramienta es capaz de dar soporte a:

Prácticas definidas en el proceso de desarrollo de software organizacional

Prácticas implementadas en los proyectos

- **La herramienta debe permitir modelar procesos de desarrollo de software para una gran variedad de tipos de proyecto y estilos de desarrollo – Mandatorio**

Tal como lo confirmaran las evaluaciones descriptas con anterioridad, en una organización conviven proyectos de todo tipo, con clientes diferentes, aplicando metodologías de desarrollo de software también diferentes. De esta forma surge la necesidad de dar soporte a esta realidad al momento de definir los procesos de desarrollo de software a nivel organizacional, siendo un

elemento fundamental el soporte que la herramienta de modelo proporcione. De esta forma es deseable que una misma herramienta permita la posibilidad de dar soporte a una variedad amplia de tipos de proyecto y estilos de desarrollo.

- **La herramienta debe estar soportada por un metamodelo – Alto valor**

Los metamodelos proporcionan la base para la definición de los modelos de procesos de desarrollo de software definidos dentro de una organización. De esta forma es una característica a considerar el hecho de que la herramienta esté soportada por un metamodelo definido.

- **La herramienta debe proporcionar mecanismos que faciliten la publicación y distribución de los procesos definidos – Alto valor**

Un punto fundamental que se debe tener en cuenta al momento de definir un proceso de desarrollo de software es que éste esté disponible a cada uno de los interesados y que cada uno de estos interesados tenga acceso a la información que efectivamente necesita. De esta forma, es una característica valorable de la herramienta el proporcionar alternativas de publicación con el fin de facilitar el acceso a la definición del proceso, pudiendo ser distribuida a todos los interesados.

- **La herramienta debe proporcionar soporte a la mejora continua de la definición de procesos y su evolución a lo largo del tiempo – Alto valor**

La definición de procesos de desarrollo de software no es algo estático sino todo lo contrario, es algo dinámico que evoluciona con el tiempo, cambia en función de las necesidades de la organización y de los proyectos, estando directamente asociado al concepto de mejora continua y al incremento en la madurez de las prácticas especificadas. La herramienta debe acompañar esta evolución proporcionando mecanismos que soporten un manejo de versiones apropiado de los procesos de desarrollo y sus componentes.

- **La herramienta debe dar soporte a las necesidades de los diferentes usuarios interesados – Bajo valor**

En la especificación de un proceso de desarrollo de software participan diferentes personas que tienen diferentes roles y por la tanto sus necesidades y las exigencias también difieren. Por ejemplo, la información que le interesa a un ingeniero de procesos, y los cambios que puede éste

hacer, son diferentes a los de un líder de proyecto o un desarrollador de software. La herramienta debe tener en cuenta esta realidad y dar soporte a esta variedad de perfiles de usuario.

- **La herramienta debe proporcionar la menor cantidad de restricciones de uso y distribución – Alto valor**

Este punto está directamente asociado a las licencias de uso requeridas para utilizar y distribuir la herramienta. Es altamente deseable que la herramienta presente las menores restricciones de uso y con la menor erogación de dinero posible.

- **La herramienta debe permitir que las tareas de modelado sean lo más simple y amigables al usuario posibles – Alto valor**

La premisa básica es que la herramienta permita al usuario llevar a cabo las tareas de modelado en el tiempo más corto posible, utilizando elementos gráficos y componentes de ayuda, intentando reducir al mínimo la necesidad de configurar archivos de texto o ejecutar líneas de comando en consola.

- **La herramienta debe permitir crear e incorporar extensiones personalizadas – Bajo valor**

Una característica deseable en una herramienta que brinda soporte al desarrollo de software en general y a la definición de sus procesos en particular, es la posibilidad de generar extensiones personalizadas, cuya finalidad es la de realizar tareas particulares que ayuden en el proceso de modelado. Un ejemplo de esto es la creación de una extensión que permita realizar una publicación automática del proceso a diferentes servidores de recursos, que representa la fase de entrega de dicho proceso. Si esta funcionalidad no estuviera disponible por defecto en la herramienta, es altamente deseable la posibilidad de que los desarrolladores la incorporen.

c. Herramientas de definición de procesos analizadas

A continuación se presentaran las herramientas de definición de procesos evaluadas. Estas herramientas fueron preseleccionadas basadas en su disponibilidad para ser probadas, considerando adicionalmente la posibilidad de que empresas de la industria local también tengan acceso a las

mismas, de manera que los resultados obtenidos puedan ser compartidos y distribuidos a dichas empresas.

"…Estas herramientas fueron preseleccionadas considerando la posibilidad de que empresas de la industria local también tengan acceso a las mismas, de manera que los resultados obtenidos puedan ser compartidos…"

Igualmente se tuvo en consideración el criterio de que en el proceso de selección participen herramientas de acceso libre y también que requieran la necesidad de disponer de licencias pagas de uso. Así, las herramientas incluidas en la evaluación fueron: Eclipse Process Framework Composer (Haumer, 2007), Team Foundation Server (MSF Team, Microsoft, 2002) orientado al modelado de procesos de desarrollo y Rational Team Concert (IBM). En la sección de anexo 1 se encuentra una descripción a cada herramienta.

d. Evaluación de Herramientas de definición de procesos

En la tabla 1 se presentan los resultados arrojados para el análisis realizado a las herramientas de modelado de procesos de desarrollo de software. Puede observarse que Eclipse Process Framework Composer (EPF) representa la opción cuyo valor cuantitativo es superior, por lo cual el proceso de selección indica que esta herramienta sería la más adecuada para definiciones de procesos.

"…Eclipse Process Framework Composer (EPF) representa la opción cuyo valor cuantitativo es superior, por lo cual el proceso de selección indica que esta herramienta sería la más adecuada para definiciones de procesos…"

Luego en los anexos se presentan en detalle los comentarios y evaluaciones de cada uno de los requerimientos planteados para cada una de las herramientas evaluadas.

Tabla 1: Resumen de análisis por herramienta

Requerimiento(R_i)	Valor cuantitativo asignado al cumplimiento del requerimiento (CR_i)	Valor cuantitativo de cada herramienta en el requerimiento (VR_{ij})		
		EPF	TFS	RTC
1	5	5	2	2
2	5	5	5	5
3	3	5	0	0
4	3	5	2	5
5	5	5	5	5
6	3	5	0	5
7	3	2	5	0
8	3	0	0	0
9	1	5	5	5
10	3	5	0	0
11	3	5	5	0
12	1	5	5	5
Valor cuantitativo general por herramienta		166	106	100

5. Selección de herramientas de simulación de procesos

En la anterior sección se llevó a cabo la selección de la herramienta de definición de procesos, en este apartado se presentara la evaluación realizada al software SimSE en el trabajo "A Quantitative Assessment Method for Simulation-based E-learnings" (Andriano, Garay Moyano, Bertoni, & Rubio, 2011), el cual será tomado como base para la fundamentación de la elección de la herramienta.

a. SimSE

Antes de profundizar en el método de evaluación de e-learnings se presentara la herramienta analizada. SimSE (Oh Navarro & van der Hoek) fue creado por la Escuela de Información y Ciencias de la Computación de la Universidad de California (Donald Bren School of Information and

Computer Sciences) y es una aplicación en que permite la creación y simulación de procesos de ingeniería de software.

"…SIMSE permite a los estudiantes a participar virtualmente en un proceso de ingeniería de software real, que involucra componentes del mundo real que no están presentes en los proyectos de clase típicas…"

Permite a los estudiantes a participar virtualmente en un proceso de ingeniería de software real, que involucra componentes del mundo real que no están presentes en los proyectos de clase típicas, tales como equipos de personas, proyectos a gran escala, toma de decisiones, personal, múltiples grupos de interés, presupuestos, planificación, y, acontecimientos inesperados al azar.

Imagen 1: Captura de pantalla de SimSE

De este modo, se ofrece a los estudiantes una plataforma en la que pueden experimentar diferentes aspectos del proceso de software de una manera práctica sin el énfasis primordial en la creación de entregables.

SimSE se dirige directamente a las deficiencias de los enfoques existentes en el aula por llenar el vacío entre el conocimiento de los procesos y la teoría impartida en las clases teóricas y el pequeño porcentaje de estos conocimientos que los estudiantes realmente poner en práctica en proyectos de clase.

b. Método de evaluación de e-learnings basados en simulaciones

Este método está basado en una exploración bibliográfica para determinar las características que deben cumplir los e-learnings y simulaciones según diferentes autores para ser considerados de buena calidad. Como resultado de la exploración se definieron categorías de evaluación que agrupan un conjunto de atributos que tienen características comunes y se definen teniendo en cuenta los componentes de los e-learnings (contenidos, LMS y LCMS) y las características clave de los simuladores (entidades del mundo real que se modelan y la experimentación que se llevará a cabo).

Tabla 1: Descripción de categorías y subcategorías de atributos de E-Learnings

Categoría	Subcategoría	Descripción
Contenido	Personalización	Se refiere a los formatos de entrega, archivos PDF, apariencia consistente, redacción técnica. Se evalúa: texto, tablas, multimedia.
	Usabilidad	Se refiere a los elementos que el usuario final tiene para manipular mejor el software. En esta categoría se evalúa: enlaces, hojas de estilo, el diseño, los controles de navegación y los mecanismos de búsqueda.
	Medición de objetivos	Se refiere a la evaluación de la forma en que los objetivos de entrenamiento y el tipo de audiencia se definen.
	Estructura Lógica	Se refiere a la evaluación de la forma en que el entrenamiento se estructura, por ejemplo, si está claramente dividido en módulos, el tiempo estipulado para cada uno de los sub módulos, las actividades que deben realizarse en cada módulo.
	Estrategias de Entrenamiento	Se refiere a las estrategias que se utilizan en el entrenamiento: tutorías, foros, grupos de trabajo, aprendizaje colaborativo, estudios de casos, debates, conferencias, capacitación para el autocontrol.
	Retroalimentación	Se refiere a la rapidez y la eficacia es la retroalimentación dada al estudiante, y evalúa la capacidad del entrenamiento para obtener retroalimentación de los estudiantes.
	Evaluación de Progreso	Se refiere a los niveles de evaluación de la Kirkpatrick. En este caso sólo evaluamos los primeros 2 niveles, los siguientes son para ser evaluados dentro del entorno de trabajo del estudiante.
	Motivación	Se refiere a la evaluación de la forma en que el entrenamiento mantiene la motivación del estudiante para terminarlo. Se evalúa tanto la comunicación síncrona y asíncrona

	Incremento de Dificultad	Se refiere a la evaluación de la capacidad de del entrenamiento para aumentar la dificultad cuando el estudiante avanza en los conocimientos adquiridos
LMS		Se refiere a la forma en que el entrenamiento administra el aprendizaje y competencias. Se evalúa: Normas, configuración del módulo, Hardware, los procedimientos de registro, informes de rendimiento estudiantil.
LCMS		Se refiere a la evaluación de la forma en que se presenta el entrenamiento, es decir: los formatos (Ej.: HTML Java, Flash, PDF), tipos de multimedia (por ejemplo: multimedia incrustados, animaciones)

Las categorías se subdividen en subcategorías más manejables, tales como: objetivos cuantificables, estructura lógica, captación, etc. y luego subcategorías fueron divididos en atributos cuantitativos. Los atributos son declaraciones expresadas como preguntas que facilitan la evaluación del software. Para este fin se define un total de diecisiete categorías y ciento sesenta y cinco atributos. Una escala logarítmica fue seleccionada para especificar prioridades de los atributos de la siguiente manera: 1 el más bajo, 3 medio y 9 la más alta. La priorización da un peso relativo en función de la importancia del atributo y se utilizará para calcular el resultado de la evaluación final.

Tabla 2: Descripción de categorías y subcategorías de atributos de Simuladores

Categoría	Subcategoría	Descripción
Experimentación	Complejidad	Se refiere a los elementos que el entrenamiento tiene para hacer el aprendizaje más adecuado a las necesidades específicas de un estudiante. Un ejemplo de tal configuración es una interfaz donde todos los parámetros esenciales se pueden modificar de acuerdo con el conocimiento anterior y experiencias que el estudiante posee.
	Retroalimentación	Ídem atributos e-learning. La diferencia es que la información obtenida está enfocada desde el punto de vista de simuladores.
Mundo Real	Contenido y conocimiento	Se refiere a la información que los simuladores dan al estudiante con el fin de ayudarle a tomar la decisión correcta.
	Realismo	Se refiere a los elementos del mundo real que se representan. Por ejemplo: los empleados, planes, plantillas, clientes.
Misceláneas		Se refiere a las cuestiones relacionadas con el entrenamiento, capacidad de instalación, la documentación, la velocidad de respuesta del simulador.

Usando este método se realizó una evaluación a la herramienta SimSE en la que se demostró que la herramienta satisface y, en algunos casos, excede los requerimientos que son evaluados. Por ello SimSE es la herramienta de simulación de procesos de software elegida a los fines del trabajo de tesis que se introduce en la especialidad.

6. Conclusiones y Trabajos Futuros

En las secciones anteriores se presentaron dos métodos que permitieron la selección de las herramientas de definición de procesos (EPF) y de simulación (SimSE). Gracias a estas herramientas se cumplen las precondiciones necesarias para la implementación de una simulación de procesos de desarrollo de software a partir de la especificación del mismo. En primer lugar EPF nos ayuda a crear una definición de procesos reusable, completa, orientada a los diferentes roles del equipo y que puede ser interpretada programáticamente. SimSE por su parte permite crear definiciones de procesos que se enfocan en los roles, artefactos y el flujo de tareas del proceso pero a diferencia de EPF estas definiciones tienen menos detalle explicativo ya que el objetivo de SimSE es que estas definiciones puedan ser ejecutadas como un juego de simulación debido a que son la entrada de su motor de generación de juegos. Teniendo en cuenta estas dos herramientas, para cumplir el objetivo final del trabajo de tesis todavía es necesario definir un modelo de simulación que tome la información de la definición del proceso para generar reglas de ejecución para un juego, y además, que transforme los conceptos del proceso definidos en EPF a sus homónimos en la definición de SimSE. Para ello en el proyecto de investigación se planteó un primer trabajo, publicado en (Bertoni, Andriano, & Rubio, 2012), en el cual se definió un conjunto básico de reglas que gobiernen la performance de la ejecución de un proceso en las dimensiones de completitud y calidad de los artefactos generados. Además, teniendo en cuenta la decisión la selección de SimSE, se definieron manualmente las reglas en SimSE para poder probar la ejecución de un proceso guiado por las reglas definidas. Posteriormente se trabajó en determinar el mapeo de los conceptos (roles, entradas, salidas, tareas, correlación de tareas, asignaciones) que describen los procesos en EPF con los correspondientes en SimSE obteniendo así una abstracción de definición de procesos que es compatible con los dos modelos de base. En la actualidad se está trabajando en la implementación de la primera prueba de concepto del software generador de e-learning que automatiza el conocimiento

antes adquirido. Esta prueba de concepto parte de una definición de procesos de EPF y genera un juego SimSE basado en la definición demostrando la hipótesis planteada: *"es posible el desarrollo de un generador automático de e-learning que sea capaz de interpretar un proceso previamente definido"*.

En el futuro se continuara con el desarrollo del modelo de base y la herramienta para luego ejercitarla en un entrenamiento del proceso de una empresa de la industria local.

7. Anexo 1: Herramientas de definición de procesos

i. Eclipse Process Framework (EPF).

EPF es una herramienta de código abierto desarrollada por el proyecto Eclipse, con capacidades que permiten la creación de definiciones de procesos y su posterior publicación. La primera versión de EPF fue desarrollada por IBM y donada Eclipse para continuar su evolución.

Esta herramienta está orientada a ingenieros de procesos, líderes de proyecto y gerentes de programas que son responsables de definir y mantener los procesos de desarrollo de la organización o de un proyecto en particular.

Comúnmente, las definiciones de procesos se enfrentan a dos problemas:

Los desarrolladores necesitan entender los métodos y practicas claves de desarrollo de software aplicadas en el proceso.

Los equipos de desarrollo tienen que definir como aplican las prácticas y métodos durante el ciclo de vida del proceso.

EPF tiene 2 objetivos que apuntan a resolver los problemas nombrados con anterioridad:

- Proveer a los desarrolladores y/o ejecutores del proceso una biblioteca de contenido que puede contener por ejemplo explicaciones de métodos, definiciones de buenas prácticas, entrenamientos, regulaciones o políticas internas y cualquier otra descripción de cómo desarrollar software. Este contenido puede referenciarse luego desde los procesos de desarrollo que acomodan las prácticas y métodos en un ciclo de vida. EPF es en parte un sistema de administración de

contenido que provee una interfaz común de acceso y administración, en contraparte de un sistema de administración de documentos que contiene un conjunto de documentos de diferentes formatos y estructura.

- Proveer capacidades de ingeniería de procesos a través de ofrecer herramientas para seleccionar y ensamblar procesos a medida de las necesidades de proyectos de desarrollo particulares. También provee bloques de construcción de procesos llamados patrones de capacidad que representan las mejores prácticas de una disciplina específica, tecnología o un paradigma de desarrollo. Estos bloques de desarrollo forman un conjunto de herramientas que permiten el rápido ensamblado de un nuevo proceso basado en las necesidades de un proyecto.

ii. Team Foundation Server

Comúnmente abreviado TFS, es un producto proporcionado por Microsoft que ofrece control de código, colección de datos, reporte, y seguimiento de proyecto y está dirigido a proyectos de desarrollo de software. Está disponible como un software stand-alone, o como la plataforma del servidor back end para Visual Studio Team System (VSTS). TFS permite a los equipos de desarrollo elegir qué metodología ellos desean usar. Inicialmente viene con dos plantillas de proceso:

MSF for Agile Software Development

MSF for CMMI Process Improvement

Cada plantilla de proceso tiene un conjunto de adaptaciones que representan la definición del proceso de desarrollo establecida en forma particular sobre la base de las plantillas disponibles. Adicionalmente se pueden incorporar plantillas proporcionadas por terceras partes o desarrollar las propias dependiendo de las necesidades (Team Center para Team Foundation Server, 2010)

iii. Rational Team Concert (RTC)

Es parte de un set de aplicaciones de gestión de ciclo de vida de proyecto (Application Lifecycle Management - ALM). En particular, RTC provee un entorno de desarrollo colaborativo proveyendo herramientas para planificación, gestión de código fuente, gestión de ítems de trabajo y gestión de builds, todo esto integrado con módulos de gestión de procesos y generación de reportes.

RTC proporciona por defecto un plugin de Scrum, pero ofrece la habilidad de modificar la implementación del proceso acorde a las necesidades de un proyecto o equipo de trabajo (IBM).

8. Anexo 2: Detalle de evaluación a herramientas de definición de procesos

A continuación se presentan en detalle los comentarios y evaluaciones de cada uno de los requerimientos planteados para cada uno de las herramientas evaluadas.

Eclipse Process Framework Composer (EPF)

Tabla 2: Análisis de Eclipse Process Framework Composer (EPF)

Req.	Comentarios adicionales
1	EPF proporciona la posibilidad de crear tanto definiciones genéricas de componentes de procesos (Method Contents) como procesos propiamente dichos (Process). De esta forma el cumplimiento de este requerimiento es completo.
2	EPF proporciona la posibilidad de partir de definiciones de contenidos de método (Method Content) y procesos (Procesos) previamente definidos, para efectuar cambias sobre los mismos, o integrando partes individuales de cada uno de ellos para generar una definición adaptada acorde con las necesidades de la organización y de los proyectos. De esta forma el cumplimiento de este requerimiento es completo.
3	La existencia de Method Content está directamente asociada con este requerimiento. Dichos Method Contents son utilizados para generar componentes genéricos y reusables, capaces de ser extendidos o utilizados tal cual están especificados. De esta forma el cumplimiento de este requerimiento es completo.
4	EPF está soportado por el metamodelo SPEM que establece una separación clara entre lo que representación la definición de componentes de proceso genéricos (Method Content) y su implementación en una organización o proyecto en término de una especificación de proceso (Process). De esta forma el cumplimiento de las exigencias planteadas en el requerimiento es completo.
5	Uno de los puntos fuertes de SPEM, es su capacidad de ser utilizado para diferentes tipos procesos de desarrollo de software, diferentes metodologías y ciclos de vida. EPF ha sido concebido con el fin de proporcionar una herramienta para poder especificar modelos basados en SPEM, siendo compatible 100%. Esto determina el cumplimiento completa del requerimiento por parte de EPF.
6	Como se expresó previamente EPF está basado en SPEM y proporciona un soporte completo a su definición. De esta forma el requerimiento es satisfecho en forma completa.
7	EPF proporciona mecanismos de exportación de las definiciones de métodos y procesos en diferentes formatos (plugin, xml, etc) dependiendo de las necesidades de los usuarios y las características de los componentes desarrollados. Un punto a considerar es que no se dispone de un mecanismo directo para que estos productos exportados sean accesibles en forma directa y simultánea por los diferentes usuarios autorizados, por ejemplo a través de su publicación en un servidor de recursos. De esta forma, EPC proporciona un cumplimiento parcial de este requerimiento.

8	EPF no proporciona un mecanismo directo para el manejo de las diferentes versiones de los componentes de procesos desarrollados. De esta forma es necesario disponer del soporte de alguna herramienta y de las prácticas de gestión de configuración para cumplimentar este requerimiento. De esta forma, EPC no proporciona un cumplimiento de este requerimiento en forma directa.
9	SPEM en su definición proporciona la posibilidad de generar vistas de los componentes de proceso que se están desarrollando. Estas vistas están definidas en término de plugins. EPF implementa este concepto de EPF para dar soporte a las necesidades de los diferentes interesados en los componentes de proceso desarrollados. Esto determina que el cumplimiento del requerimiento sea completo.
10	EPF está desarrollado sobre la base de la plataforma Eclipse, lo cual permite su libre utilización, sin necesidad de disponer de licencias ni alguna otra restricción al momento de usarla y generar especificaciones de componentes de procesos propios. De esta forma el cumplimiento de este requerimiento es completo.
11	En cierta forma difícil de calificar este requerimiento asumiendo que diferentes personas pueden tener diferentes opiniones acerca de si una herramienta es fácil de usar, y más aún el concepto de agradable, directamente asociados a los gustos y expectativas particulares. Sin embargo, considerando un nivel de expectativas básicas de los diferentes usuarios, EPF resulta una sencilla de usar una vez que se dispone de cierta experiencia y práctica trabajando con este tipo de ambientes de desarrollo. Se asigna un cumplimiento completo a este requerimiento.
12	Al estar desarrollado sobre la plataforma Eclipse, EPF proporciona la posibilidad de que desarrolladores sean capaces de generar sus propios plugins (no confundir con los plugins de SPEM) para adicionarlos a la plataforma y dar soporte a necesidades particulares, proporcionando alternativas de customización. Igualmente estos plugins pueden ser exportados a todos los ambientes de EPF que sean compatibles. Esto determina un cumplimiento completo del requerimiento.

Team Foundation Server (TFS)

Tabla 3: Análisis de Team Foundation Server (TFS)

Req.	Comentarios adicionales
1	TFS proporciona por defecto dos plantillas de procesos ya definidas, una para metodologías ágiles y otra para procesos CMMI, pero la creación de plantillas adicionales es responsabilidad de empresas de tipo partner. Por otra parte, la edición de plantillas ya existentes requiere de herramientas externas, como InfoPath (propiedad de Microsoft). Esta limitación para la creación de plantillas "from scratch" es valorizada como un cumplimiento parcial.
2	TFS ofrece la habilidad de modificar plantillas ya creadas (CMMI y Agile). Se debe tener en cuenta que para hacerlo se requiere de herramientas externas, entre ellas InfoPath. Esto determina el cumplimiento absoluto del requerimiento.
3	No existe en TFS el concepto de "componentes de procesos" reusables. Todas las definiciones de procesos parten de una base de 5 fases, que se asume se repetirá en los distintos procesos, pero no define prácticas reutilizables en los distintos procesos.
4	En TFS las plantillas creadas para procesos CMMI y Agile están fuertemente ligadas a la implementación de los mismos en una organización/proyecto. Sin embargo, ofrece muchas facilidades para, da una implementación particular y adaptarla a las necesidades del proyecto. Por lo tanto se asigna un cumplimiento parcial del requerimiento.
5	A través de la utilización y adaptación de diferentes plantillas, TFS da soporte a cualquier tipo de proyecto de desarrollo de software. Cumplimentando en forma

	absoluta el requerimiento.
6	TFS está basada en MSF, que surgió como un compendio de guías para el diseño, desarrollo, implementación y soporte de soluciones de una manera efectiva, basadas en tecnologías proporcionadas por Microsoft, lo cual no representa en sí mismo un metamodelo. De esta forma no se cumplimenta el requerimiento.
7	TFS provee facilidades de publicación de los procesos definidos, de manera que puedan ser accedidos por los distintos stakeholders. Por otra parte, permite distribuir las definiciones de procesos en formato XML, así como también los instaladores para nuevas plantillas de procesos. Esto determina el cumplimiento absoluto del requerimiento.
8	TFS no proporciona un mecanismo directo para el manejo de las diferentes versiones de los componentes de procesos desarrollados. Es necesario disponer del soporte de alguna herramienta de administración de configuración y de las prácticas de gestión de configuración para cumplimentar este requerimiento. Por otra parte, como ciertas plantillas son creadas por terceros, muchas veces será necesario a que estos grupos ofrezcan las actualizaciones correspondientes.
9	TFS ofrece distintas vistas de la información, de acuerdo a las necesidades de cada usuario, lo que determina el cumplimiento absoluto del requerimiento.
10	Si bien MSF es gratuito pues surgió de una compilación de las mejores prácticas de desarrollo de software de Microsoft, TFS no es de uso libre. De la misma forma, la creación de nuevas plantillas está principalmente delegada a empresas especializadas que establecen convenidos para tal fin. Esto determina el no cumplimiento del requerimiento de acuerdo a las necesidades del proyecto de investigación
11	Este es definitivamente un punto fuerte en Microsoft. Más allá de que un usuario puede editar los archivos XML directamente, las herramientas disponibles (como InfoPath) permiten la edición de las plantillas de un modo intuitivo y amigable al usuario.
12	TFS está pensado para funcionar embebido en el entorno de desarrollo de software de Microsoft. Los desarrolladores pueden, entonces, valerse de las características propias del IDE y de los lenguajes de programación para crear fácilmente nuevos plugins que puedan ser adicionados a la herramienta.

Rational Team Concert (RTC)

Tabla 4: Análisis de Rational Team Concert (RTC)

Req.	Comentarios adicionales
1	RTC admite la implementación de distintos procesos. Sin embargo, la definición de un nuevo proceso no es trivial, y hasta el momento las definiciones de procesos son solamente proporcionadas por los equipos de desarrollo de IBM (Jazz).
2	Existen dos niveles de personalización de procesos, una a nivel de proyecto y otra a nivel de equipo. Cada proyecto creado toma una definición de procesos inicial y puede realizar cambios sobre el mismo. Esos cambios se van a ver reflejados para todos los equipos debajo de ese proyecto. Cada team puede a su vez modificar el proceso acorde a sus necesidades.
3	Los procesos se implementan como un todo. No es posible, con RTC, definir módulos independientes que puedan ser reutilizados por uno o más procesos. Cada proceso se define como un todo, y cada implementación del mismo puede ser modificada a nivel de proyecto y/o a nivel de equipo. Pero una vez modificado, todo cambio en el

	proceso original no se refleja en las instancias adaptadas.
4	Dada una definición de procesos, cada proyecto puede implementarla y traspasar esa misma definición a sus equipos.
5	RTC permite la inclusión de múltiples procesos. Cada vez que se crea un proyecto, se puede elegir entre cualquiera de los procesos disponibles.
6	La implementación de RTC está basada en SPEM.
7	Con esta herramienta no se puede publicar una definición de procesos. Esta definición sólo es accesible a través del lenguaje fuente (XML). Por otra parte, no es posible definir diferentes vistas del proceso acorde a los roles de cada usuario, sino que existe una sola vista.
8	RTC no maneja versionado de procesos. Una vez que se realizaron cambios en los mismos, no es posible retornar a una versión previa.
9	Dada la implementación de un proceso particular en un proyecto o equipo, RTC ofrece diversas vistas, de acuerdo al perfil de cada usuario y por ende acorde a sus necesidades.
10	RTC es una herramienta paga, bajo licencia de IBM.
11	Actualmente, la definición y personalización de los procesos se hace mediante la edición de un XML.
12	Al ser una aplicación basada en la plataforma Eclipse, es posible desarrollar plugins para satisfacer necesidades adicionales/particulares de un proyecto o equipo.

Bibliografía

Aldrich, C. (2009). *Learning Online with Games, Simulations, and Virtual Worlds: Strategies for Online Instruction (Jossey-Bass Guides to Online Teaching and Learning).*

Ambler, S. W., & Holitza, M. (2012). Getting Started with Agile. En S. W. Ambler, & H. Matthew, *Agile for Dummies* (págs. 15-24). New Jersey: John Wiley & Sons, Inc.

Andriano, N., Garay Moyano, M., Bertoni, C., & Rubio, D. (2011). *A Quantitative Assessment Method for Simulation-based E-Learnings.* IEEE-CS Conference on Software Engineering Education and Training.

Benbow, D. W. (2005). *The Certified Six Sigma Black Belt Handbook.* ASQ.

Bertoni, C., Andriano, N., & Rubio, D. (2012). Generación y Actualización automática de un e-learning basado en una definición de proceso SPEM-compatible. CACIC.

CMMI Product Team. (2006). *CMMI for Development, version 1.2.* Software Engineering Institute.

Cohn, M. (2009). An Overview. En M. Cohn, *User Stories Applied for Agile Software Development* (págs. 3-15). Indiana: Addisson - Wesley.

Cohn, M. (2009). *User Stories Applied: For Agile Software Development.* Indiana: Addison Wesley.

Cohn, M. (2009). What stories are not. In M. Cohn, *User Stories Applied for Agile Software Development* (pp. 133-144). Indiana: Addison-Wesley.

Cohn, M. (2009). Why User Stories? In M. Cohn, *User Stories Applied for Agile Software Development* (pp. 145-155). Indiana: Addison-Wesley.

Cohn, M. (s.f.). *Mountain Goat Software*. (Mountain Goat Software) Recuperado el 1 de Diciembre de 2012, de User Stories: http://www.mountaingoatsoftware.com/topics/user-stories

Corporation, M. (2010, 04 29). *Team Center para Team Foundation Server. MSDN.* . Retrieved from http://www.microsoft.com/spanish/msdn/latam/vstudio/teamsystem/team/

Corral, R. (12 de Noviembre de 2007). *Exprimiendo Scrum: Scrum y la gestión de requisitos*. (Los pensamientos, peleas y descubrimientos de Rodrigo Corral con Scrum, C++, C#, ASP.Net, Team System, Sql Server, Sharepoint, la arquitectura, la gestión de proyectos y el desarrollo de software en general...) Recuperado el 6 de Junio de 2012, de http://geeks.ms/blogs/rcorral/archive/2007/11/12/exprimiendo-scrum-scrum-y-la-gesti-243-n-de-requisitos.aspx

D. Rubio, N. A. (2008). *An integrated improvement framework for sharing assessment lessons learned.* La Rioja: Proceedings del XIV Congreso Argentino de Ciencias de la Computación.

DoD. (1998). *DoD Modeling and Simulation (M&S) Glossary.* DoD.

Donald Bren School of Information and Computer Sciences. (n.d.). Retrieved from http://www.ics.uci.edu/

Figuerola, N. (Febrero de 2012). *Definiendo Requerimientos: Tradicional vs. Caso de Uso vs. Historias de Usuario.* Recuperado el 16 de Febrero de 2013, de http://articulosit.files.wordpress.com/2012/04/requerimientos.pdf

Firesmith, D. G. (s.f.). *Use Cases: the Pros and Cons*. Recuperado el 14 de Diciembre de 2012, de Knowledge System Corporation: http://www.ksc.com/articles/usecases.htm

González, C. J., Izaurralde, M. P., Marzo, L. G., & Rubio, D. M. (2009). *Experiencia de la Aplicación de Aprendizaje Activo en un Marco Universidad-Empresa*, . TEyET.

Haumer, P. (2007, 04). *Eclipse Process Framework Composer - Part 1Key Concepts.* Retrieved 04 29, 2010, from http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf

Henderson, A. J. (2003). *The E-Learning Question and Answer Book: A Survival Guide for Trainers and Business Managers.*

Humphrey, W. S. (1989). *Managing the software process.* Addison-Wesley.

Humphrey, W. S. (1989). *Managing the Software Process.* Massachusetts: Addison-Wesley.

IBM. (1998). *Rational Unified Process Best Practices for Software Development Teams.* Retrieved from IBM developerworks: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

IBM. (n.d.). *Rational Team Concert*. (Rational Software) Retrieved 04 29, 2010, from http://www.ibm.com/developerworks/rational/products/rtc/

J.L.Pfleeger. (2002). *Ingeniería del Software: Teoría y Prática.* Buenos Aires: Prentice Hall.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.

Kaczor, K. (3 de Agosto de 2011). *5 Common Misktakes We Make Writing User Stories.* Recuperado el 1 de Diciembre de 2012, de Scrum Alliance: http://www.scrumalliance.org/articles/366--common-mistakes-we-make-writing-user-stories

Langenfeld, C. (8 de Julio de 2011). *Using Rally to Map High Traceability User Stories: PRD to SRS.* Recuperado el 20 de Enero de 2013, de Rally Blogs: http://www.rallydev.com/community/agile-blog/using-rally-map-high-traceability-user-stories-prd-srs

Laycock, M. (2005). *Collaborating to compete: achieving effective knowledge sharing in organizations.* The Learning Organization: Emerald Group Publishing Limited, 2005.

Leffingwell, D., & Behrens, P. (2009). *A User Story Primer.* Recuperado el 24 de Enero de 2013, de http://trailridgeconsulting.com/files/user-story-primer.pdf

Lin, H., Wen-Hsiang, S., Zhi-Wei, Y., & Don-Lin, Y. (2008). Applying UML and software simulation for process definition, verification, and validation. *Information and Software Technology*, *50*(9-10), 897-911.

Mauricio Silclir, P. S. (2010). Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software. JAIIO.

MIT. (2012). *http://www.mit.edu/*. Retrieved 2012, from http://www.mit.edu/

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

Munro, R. A. (2005, May). Understanding the Buzz Around E-Learning: Searching for Faster/Better/Cheaper Learning - Effectiveness of E-Learning Techniques. *ASQ World Conference on Quality and Improvement Proceedings, 59*(0), 131-143.

Natalia Andriano, M. G. (2011). A Quantitative Assessment Method for Simulation-based E-learnings.

NTL Institute. (2012). *NTL Institute*. Retrieved from http://www.ntl.org/

Oh Navarro, E., & van der Hoek, A. (n.d.). SimSE An Interactive Simulation Game for Software Engineering Education.

OMG. (2008). *Software & Systems Process Engineering Metamodel Specification (SPEM).*

OMG. (2010). *UML 2.3 Specification.* OMG.

P. Szyrko, M. S. (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.

P. Szyrko, M. S. (2009). Un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software. WICC.

Portela, C., Vasconcelos, A., Silva, A., Sinimbú, A., Silva2, E., Ronny, M., et al. (2012). *A Comparative Analysis between BPMN and SPEM Modeling Standards in the Software Processes Context.* Journal of Software Engineering and Applications.

Roberto, M. A. (2009). *Leadership Lessons From The Sims.* Retrieved 6 21, 2012, from http://clomedia.com/articles/view/leadership_lessons_from_the_sims

Schwabe, K. (2004). *Agile Project Management with Scrum.* Microsoft Press.

Shank, C. (2001). *Designing Word-Class E-Learning, How IBM, Hardvard Bussiness School and Columbia University Are succeding at E-Learning.*

Shannon, R. E. (1975). *System Simulation: The Art and Science.*

Software Engineering Institute. (2010). *CMMI for Development, Version 1.3.* Massachusetts: CMMI Product Team.

Software Engineering Standards Committee of the IEEE Computer Society. (25 de Agosto de 1998). *IEEE Recommended Practice for Software Requirements Specifications.* Recuperado el 6 de Diciembre de 2012, de IEEE Std 830-1998: http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

Squire, K., Barnett, M., M. Grant, J., & Higginbotham, T. (2004). *Electromagnetism supercharged!: learning physics with digital simulation games.* ACM.

Stenning, V. (1987). *On the role of an environment.* California : IEEE Computer Society Press.

Suscheck, C. (17 de Enero de 2012). *Defining Requirement Types: Traditional vs. Use Cases vs. User Stories.* Recuperado el 20 de Enero de 2013, de Agile Techwell: http://agile.techwell.com/articles/weekly/defining-requirement-types-traditional-vs-use-cases-vs-user-stories

Tague, N. N. (2004). *The Quality Toolbox, Second Edition.* ASQ Quality Press.

Tague, N. R. (2004). *The Quality Toolbox.* ASQ.

Team Center para Team Foundation Server. (2010). (Microsoft Corporation) Retrieved 04 29, 2010, from MSDN: http://www.microsoft.com/spanish/msdn/latam/vstudio/teamsystem/team/

Thomas, R. (2001). *Interactivity & Simulations in e-Learning.* MultiVerse Publications.

Troy University. (2012). *Troy University*. Retrieved from http://www.troy.edu/

University of Central Columbia. (n.d.). Retrieved 2012, from Institute for Simulation and Training: http://www.ist.ucf.edu/background.htm

Van Dam, N. (2004). *The E-Learning Fieldbook.* McGraw-Hill.

Varhol, P. (2012). Conference Paper: Agility with Traceability: Blending Requirements and User Stories. *Quality Engineered Software & Testing Conference & Expo.* Chicago.

Williams, L. (2004). *Agile Requirements Elicitation*. Recuperado el 20 de Enero de 2013, de http://agile.csc.ncsu.edu/SEMaterials/AgileRE.pdf