Un Método Heurístico para el Análisis y Selección de Herramientas de Modelado de Procesos de Desarrollo de Software

Mauricio Silclir, Pablo Szyrko, Álvaro Ruiz de Mendarozqueta, Diego Rubio

{47920, pablo.szyrko, aruiz, drubio}@sistemas.frc.utn.edu.ar
Laboratorio de Investigación en Ingeniería y Calidad de Software
http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/
Departamento de Ing. en Sistemas de Información
Universidad Tecnológica Nacional
Maestro M. López esq. Cruz Roja Argentina
(X50165ZAA) Ciudad Universitaria, Córdoba, Argentina

Abstract. La creciente complejidad y necesidad de información relacionada a los procesos de desarrollo de software en las industrias, determina que el acceso a una herramienta que soporte la definición y mantenimiento de dichos procesos, sea un aspecto cada vez más importante.
El presente trabajo, enmarcado dentro del proyecto de investigación acerca del desarrollo de un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software [1], plantea el análisis comparativo y la selección de herramientas de modelado de proceso de desarrollo de software. Este análisis comprende la definición de un conjunto de requerimientos básicos que deben cumplir las herramientas, junto con un método heurístico de decisión para la selección de la herramienta que mejor se adapta a las necesidades de un proyecto.
A modo de implementación de los resultados de la investigación, se planteó un análisis sobre un conjunto de aplicaciones de definición de procesos, EPF, TFS y RTC, resultando la primera como herramienta más adecuada para el proyecto planteado en [1].
El mismo proceso de análisis y selección puede ser aplicado a otros proyectos, pudiendo modificarse las categorizaciones y valoraciones de los requerimientos y su cumplimiento de acuerdo a sus objetivos y necesidades particulares.

Keywords: Herramienta de modelado de procesos - Modelo - Proceso – Comparación – Requerimientos – Método Heurístico

1 Introducción

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 [2] hasta llegar a las actuales tecnologías de desarrollo [3].

La premisa de que la calidad de un producto está influenciada por la calidad del proceso que se utiliza para construirlo [4][5] determina la necesidad de que todo proyecto de software cuente con un proceso de desarrollo robusto [6] [7]. En este contexto, se genera 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 [8]. 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 [9], y proveerá a los profesionales el marco de trabajo que necesitan para llevar a cabo un trabajo profesional de manera consistente [8].

Bajo estas premisas, se presentan algunos problemas al pensar cómo definir el proceso de desarrollo en una organización [9 pág. 9]:

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 [3], 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 [10] [11]. 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 [12].

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. Sin embargo, como un paso previo, la organización necesita también delinear una estrategia de automatización que guíe el uso de nuevos métodos y técnicas. Entonces, las herramientas antes mencionadas 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 [4]. Hoy en día, hay múltiples herramientas para el modelado de procesos disponibles en el mercado, con distintas características que las distinguen una de otra. Bajo esta circunstancia, surge entonces la necesidad de contar con un método que nos permita seleccionar la herramienta más adecuada tomando como base el conjunto de requerimientos básicos identificados.

El presente trabajo establece un conjunto de requerimientos a ser cumplimentados por dichas herramientas, junto con un método heurístico de comparación y selección de las mismas con el objetivo de ayudar a las empresas en el análisis y selección de la herramienta de modelado de procesos de desarrollo de software que mejor se adapten a sus necesidades. Asimismo, presenta una comparación de herramientas actualmente disponibles en el mercado. Dicha comparación se realizó en el contexto del trabajo del proyecto de investigación sobre el Desarrollo de un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software [1].

2 Metodología de análisis y selección

El proceso formal de análisis y selección utilizado deriva de la matriz de decisión propuesta en [13]. La principal variación respecto de este modelo es la posibilidad de eliminar una de las alternativas consideradas para la decisión si esta no cumpliera con uno o más atributos identificados como mandatorio, en el marco de las herramientas de modelado de procesos.

Así, el modelo planteado en este trabajo consta de tres etapas secuenciales:

2.1 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 presente trabajo, los cuales pueden ser consultados en [1].

La segunda fuente considerada fueron las necesidades de la industria local, de acuerdo a los resultados de la investigación citada en [14]. De acuerdo a dicha investigación, durante el relevamiento realizado (se compilaron 40 evaluaciones de calidad en 14 empresas) pudo observarse que las diferentes empresas evaluadas tenían la necesidad de contar con distintos procesos de desarrollo debido a la variedad de productos y proyectos que estaban desarrollando. En ninguna de las organizaciones evaluadas pudo encontrarse un entorno automatizado del proceso definido. En dos de ellas pudo verse una herramienta para automatizar el proceso definido que contenía un flujo de trabajo con la capacidad de adjuntar documentos en cada instancia. Esos documentos son usualmente diagramas, texto y planillas de cálculo. Los entornos específicos de construcción de software no estaban integrados con el proceso definido e instanciado en cada proyecto.

Lo anterior implica que es factible asignar otras categorías aplicando el mismo conjunto de requerimientos, de acuerdo a las características y necesidades particulares de cada proyecto.

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.

2.2 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.

2.3 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:

CR_i: Categoría asignada al requerimiento i
VR_{ij}: Valoración aplicada para el requerimiento i en la herramienta j
V_j: Valoración cuantitativa de la herramienta
$V_j = \sum_{i=1..n} CR_i * VR_{ij}$

- Selección de herramienta

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

3 Alternativas analizadas

Para este trabajo se consideraron como herramientas alternativas a Eclipse Process Framework Composer[15], Team Foundation Server [16] orientado al modelado de procesos de desarrollo y Rational Team Concert (RTC) [17]. Éstas herramientas fueron preseleccionadas basadas en su disponibilidad para ser probadas en el contexto del proyecto de investigación, considerando adicionalmente la posibilidad de que empresas de la industria local también tengan acceso a las mismas, de manera que los resultados obtenidos en el proyecto de investigación puedan ser compartidos y distribuidos a dichas empresas. 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.

A continuación se presenta una breve introducción a las herramientas analizadas:

- Eclipse Process Framework Composer (EPF Composer) es una herramienta para ingenieros de procesos, líderes y administradores de proyectos, quienes son responsables de mantener e implementar procesos para organizaciones dedicadas al

desarrollo o para proyectos individuales. El lenguaje de metamodelo subyacente de EPF es SPEM2.0 (Software Process Engineering Metamodel), estándar definido por el OMG (Object Management Group) para la representación de procesos de desarrollo [17].

- 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 [18].

- 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 [17].

4 Requerimientos de las herramientas

Para brindar una definición efectiva y eficiente de un proceso de desarrollo, las herramientas de definición de proceso deben cumplir con una serie de requerimientos básicos. En líneas generales, basado en lo expuesto por Stenning en [19] (y citado por Humphrey en [4]), las herramientas deben proveer un uso cómodo y conveniente, brindar soporte para la personalización de procesos, estar basadas en una arquitectura abierta y poseer un esquema conceptual comprensible que abarque bases de datos, datos de proceso, interfaces con herramientas y entorno evolutivo [19].

En la misma dirección, Pfleeger [3] establece cinco características básicas que debe poseer cualquier herramienta de modelado y automatización de procesos, a saber: facilitar la comprensión y la comunicación humanas, dar soporte para la mejora del proceso, dar soporte para la gestión del proceso, brindar una guía automatizada en la realización del proceso y dar soporte para la ejecución automatizada de procesos.

Con base en las características y consideraciones antes mencionadas, y teniendo en cuenta la experiencia adquirida en el mercado local durante las evaluaciones descriptas anteriormente (sección 2.1), a continuación se detallan los requerimientos específicos 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 en la sección 2.1, 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.

5 Análisis de las herramientas basadas en los requerimientos

5.1 Sumarización de resultados

En la tabla 1 se presentan los resultados arrojados para el análisis realizado de las herramientas de modelado de procesos de desarrollo de software. En la siguiente sección se presentan en detalle los comentarios y evaluaciones de cada uno de los requerimientos planteados para cada uno de las herramientas evaluadas en este trabajo.

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

Nota adicional: no hay requerimientos categorizados como mandatorios que no estén cumplimentados por las diferentes herramientas analizadas, por lo cual las tres herramientas se presentan como candidatas válidas para análisis y selección.

5.1.1 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.

5.1.2 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.

5.1.3 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.

5.2 Resultado final

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 el proyecto Desarrollo de un modelo de validación automático para la definición y mantenimiento de procesos de desarrollo de software[1].

6 Conclusiones y trabajos futuros

Esperamos que en las organizaciones sea cada vez más común el uso de una herramienta que facilite la definición, ejecución y mantenimiento de sus procesos de desarrollo de software.

Debido a la diversidad de ofertas de este tipo de herramientas, es necesario establecer un criterio de selección que permita a un proyecto u organización realizar una elección adecuada, acorde con sus necesidades y objetivos, y teniendo en cuenta lo recomendado por referentes en la temática como lo son Humphrey [4] y Pfleeger[3]. Para ello, el establecimiento de un conjunto de requerimientos básicos categorizados de acuerdo a su impacto en el proyecto, juega un papel fundamental para establecer ese criterio de decisión.

A su vez, es necesario establecer un método de comparación que sea capaz de cuantificar los aspectos cualitativos de los requerimientos mencionados. De esta manera se podrá realizar una selección más objetiva de la herramienta que mejor se adecúe a las necesidades de la organización.

El método presentado en este trabajo permite que, a través de la variación de los valores asignados a los requerimientos, pueda adaptarse a otros proyectos u organizaciones, como así también se puedan modificar o actualizar los requerimientos durante la fase de adaptación del proceso de análisis y selección.

Los requerimientos definidos fueron determinados basados tanto en las recomendaciones de autores reconocidos en la temática [3][4] como así también por las

necesidades detectadas en las más de 40 evaluaciones realizadas durante este trabajo en el mercado local [14].

Para el caso del proyecto de investigación en cuyo marco se realizó el presente trabajo, el análisis se realizó en base a las herramientas EPF, TFS y RTC, y se determinó que, en el contexto del proyecto, la más conveniente es EPF.

En el presente trabajo el método ayuda a elegir cuál de todas las herramientas cumple de manera más completa con todos los requerimientos planteados, pero no permite la selección de varias de ellas. Dado que en la actualidad existen proyectos de integración entre algunas de las herramientas disponibles, se propone como trabajo futuro mejorar el método presentado a través de la posibilidad de combinar más de una herramienta como opción de selección, e incluir un análisis que determine la sinergia que podrían brindar dos o más herramientas combinadas, de acuerdo a las necesidades establecidas en el contexto en el que se está realizando el análisis.

7 Referencias

1. **P. Szyrko, M. Silclir, G. García Favre, D. Rubio.** *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.

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

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

4. **Humphrey, Watts S.** *Managing the Software Process.* Massachusetts : Addison-Wesley, 1989. ISBN: 0201180952.

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

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

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

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

9. *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.

10. **Rolland, C., Souveyet, C. and Moreno, M.** *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.

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

12. **Oliveira, Sandro et al.** *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.

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

14. **D. Rubio, N. Andriano, A. Ruiz de Mendarozqueta, C. Bartó.** *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.

15. **Haumer, Peter.** Eclipse Process Framework Composer - Part 1Key Concepts. [En línea] 04 de 2007. [Citado el: 29 de 04 de 2010.] http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf.

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

17. **IBM.** Rational Team Concert. [En línea] Rational Software. [Citado el: 29 de 04 de 2010.] http://www.ibm.com/developerworks/rational/products/rtc/.

18. Team Center para Team Foundation Server. *MSDN*. [En línea] Microsoft Corporation, 2010. [Citado el: 29 de 04 de 2010.] http://www.microsoft.com/spanish/msdn/latam/vstudio/teamsystem/team/.

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