AKADÈMEIA

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Presentación del Método KDDP para la creación de Software de Investigación

Gerardo Cerda Neumann[1]

Héctor Antillanca Espina[2]

Fecha Recepción: 3 de Noviembre de 2014.

Fecha de Aceptación: 9 de Agosto de 2017.

Resumen: Se presenta el concepto de desarrollo de software de investigación. Se caracterizan sus dificultades y se propone un método formal llamado KDDP (Knowledge DiscoveryDriven Process) como solución. Basado en el Modelo de la Espiral para gestionar el desarrollo de software dentro de una investigación que se esté llevando a cabo y soluciona la problemática del aumento de productividad de los equipos de desarrollo de software de investigación. Se utiliza el Diálogo de Bohm (descrito por el mismo autor en el año 1997 como una herramienta que destaca la primacía de la cooperación por sobre la competencia, factor fundamental para mejorar la productividad) para hacer circular el significado y compartir el conocimiento generado. Se insiste en que la realización del diálogo como apoyo a la circulación del conocimiento sea efectuado suspendiendo los juicios, compartiendo la visión de los participantes y respetando los planteamientos de cada uno de ellos. Este planteamiento se completa con la aplicación de "las cuatro movidas de Kantor", corresponden a: mover, oponer, observar y seguir las opiniones de los participantes. Cabe destacar que el método utiliza un conjunto de herramientas: reuniones diarias de coordinación de 15 minutos "de pie", una Wiki para registrar los compromisos del equipo de investigación, un generador automático de documentación de código, un sistema de control de versiones y una "Matriz de

[1] Gerardo Cerda Neumann: Magíster en Ingeniería Informática. Docente Facultad de Negocios, Ingeniería y Artes Digitales, Universidad Gabriela Mistral. Correo electrónico: gerardo.cerda@ugm.cl, Santiago, Chile

[2] Héctor Antillanca Espina: Doctor en Informática. Docente Departamento de Ingeniería Informática, Universidad de Santiago de Chile. Correo electrónico: hector.antillanca@usach.cl, Santiago, Chile

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Trazabilidad" para registrar las aplicaciones ejecutadas junto con los datos utilizados a fin de llevar un registro claro de la evolución del software que se ha construido para la investigación.

Palabras clave: software de investigación, Modelo de la Espiral, Diálogo.

Introducing the KDDP Method for the creation of investigation software

Abstract: this paper present the concept of research software development. Their difficulties are characterized and a method called KDDP (Knowledge Discovery-Driven Process) is proposed like solution. This method is based on spiral model for manage the development of software inside the investigation. Uses Bohm's Dialogue for circulating the meaning and to share the knowledge generated.

Keywords: research software, spiral model, dialogue.

1. Introducción

En los últimos años el interés en el desarrollo de software con propósito de investigación científica ha aumentado notablemente (De Roure, 2009; Morris & Segal, 2012; NguyenHoan, Flint & Sankaranarayana 2010; Segal, 2008a; Segal, 2008b; Segal, 2008c; Tan &

Phillips 2005). Se entiende como "Software de Investigación" el que es construido como parte de un proyecto de investigación. El desarrollo de este tipo de software posee características diferentes al desarrollo de software comercial (Kelly & Smith, 2010). Normalmente este tipo de software se construye en ambientes de investigación, donde los mismos integrantes del equipo de desarrollo están interesados en los productos y en general son los usuarios finales de las aplicaciones. Por esta misma razón los requerimientos son definidos por el mismo equipo que desarrolla el producto sin participación de un cliente externo (Segal, 2012). De hecho existen experiencias donde se ha tratado de transferir el desarrollo de estas aplicaciones

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

desde profesionales de la Ingeniería de Software que apoyan los proyectos al mismo equipo lo que ratifica la característica de que son aplicaciones construidas por y para ellos mismos (Fogli & Parasiliti, 2012).

En este tipo de proyectos su desarrollo está guiado por el conocimiento que se va logrando conforme ellos avanzan (Antillanca & Cerda, 2010). Esto ocurre porque cuando los proyectos se inician no está totalmente definido ni lo que se obtendrá ni el aprendizaje que se logrará con el mismo. En este contexto el desarrollo del software claramente sigue un esquema evolutivo ya que hay aproximaciones sucesivas a una versión que funcione según lo que se requiere (Sommerville, 2011). Sin embargo se presentan las mismas dificultades para cumplir los plazos de entrega, escasa documentación, difícil coordinación del equipo de trabajo y tiempo excesivo para modificar el código escrito por otros desarrolladores. En resumen se puede decir que es difícil lograr productividad en la creación del software de investigación.

2. Formulación del problema

Este tipo de software es desarrollado en el contexto de investigaciones científicas y su objetivo es apoyar el logro de estas últimas. Se ha definido en consecuencia "Software Científico" como "una aplicación que proporciona datos para apoyar decisiones en algún campo de la ciencia o ingeniería" (Kelly & Smith, 2010). Se utilizará el concepto de "software de investigación" ya que se construye en el contexto de una investigación científica. Este concepto se refiere a todo tipo de software que se construya en el marco de una investigación científica y por esa razón incluye al "Software Científico".

Del análisis de este tipo de proyectos es posible obtener ciertas características en común (Antillanca & Cerda 2010; Howison & Herbsleb 2012; Segal, 2005; Segal 2009a):

1. Muchos de los proyectos de desarrollo de software de investigación se inician por la necesidad de evaluar una técnica de procesamiento de datos a partir del análisis de un área de conocimiento específico en estudio.

2. Al ser aplicaciones desarrolladas en el contexto de la investigación el dominio del problema del proyecto es de alta complejidad.

3. La ejecución de la aplicación se denomina experimento y consiste en el procesamiento de un conjunto de datos extraídos de la misma investigación.

4. El objetivo del proyecto no es la creación del software en sí. Este se construye para realizar los experimentos propios de la investigación.

5. En algunos casos se busca la creación de un algoritmo que de una mejor solución al problema en estudio que los ya existentes. En este caso el software que se construye se crea solo para probar el algoritmo propuesto.
6. Al empezar el proyecto es difícil definir los requerimientos de la aplicación que se construirá. Los requerimientos se van clarificando conforme avanza el proyecto y se genera conocimiento.

7. Los equipos de proyecto en general son pequeños (máximo 5 o 7 integrantes), de alta capacidad técnica y cuyos integrantes están muy motivados por el proyecto.

8. La interfaz de usuario de la aplicación es sencilla ya que es utilizada por los mismos desarrolladores.

9. Los miembros del equipo participan de una cultura común que los lleva a no documentar lo que para ellos es obvio. Esto se transforma en una dificultad cuando alguien ajeno al equipo necesita entender las aplicaciones construidas.

10. En ocasiones existe la necesidad de procesar grandes volúmenes de datos lo que obliga a constantes ajustes y mejoras a las aplicaciones desarrolladas.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

11. Se generan múltiples versiones de las aplicaciones en varios computadores distintos cada uno con sus datos de ejecución.

12. Es difícil saber qué resultado se generó con qué versión de la aplicación y cuáles fueron los datos procesados.

13. Generalmente no siguen estándares ni de diseño ni de documentación.

14. Generalmente la aplicación construida queda "amarrada" a quien la codificó haciendo difícil que otra persona la modifique.

15. Muchas veces al avanzar los proyectos dan origen a otros subproyectos donde se analizan y exploran ideas surgidas del trabajo original.

16. Pueden surgir cambios bruscos en los requerimientos de las aplicaciones que se han definido lo que provoca dificultad para reenfocar los recursos y los esfuerzos del equipo de trabajo.
17. Y por último lo más importante: **los proyectos son guiados por el descubrimiento del conocimiento**.

Dentro de este tipo de software es posible mencionar aquellos que simulan motores de búsqueda, como los que se realizan en los laboratorios de investigación de empresas tales como Yahoo! y Google. También aparecen simuladores de modelos científicos donde se busca encontrar algún tipo de mejora u optimización en el funcionamiento, como los que se desarrollan en laboratorios investigación de muchas universidades del mundo. Como ejemplo se podrían considerar también aplicaciones que modelan procesos químicos, industriales y económicos.

Los hitos genéricos para este tipo de proyecto son:

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

1. Definir el problema que se desea investigar o la técnica que se desea probar.

2. Asignar los recursos necesarios al proyecto y definir quién lo liderará.

3. Hacer un estudio inicial de las técnicas que se utilizarán. Eventualmente generar algunos prototipos para probar dichas técnicas.

4. Realizar una investigación sobre los conceptos que se van a aplicar.

5. Definir el experimento que se va a realizar así como los datos que se utilizarán.

6. Construir el software que se necesita para realizar el experimento.

7. Realizar el experimento procesando los datos con el software construido y extraer conclusiones útiles.

Los pasos 5, 6 y 7 se pueden repetir varias veces hasta lograr el conocimiento suficiente para dar por terminado el proyecto.

8. Escribir el artículo a partir de los resultados obtenidos.
9. Eventualmente repetir el experimento modificando el software o los datos utilizados.

Dependiendo de los resultados se puede modificar el artículo escrito.
Una vez concluido el artículo o informe técnico se lo debe presentar a una instancia de revisión superior a la del equipo de proyecto. En algunas ocasiones es posible que se creen patentes asociadas a la investigación.

Por último cabe destacar que resulta difícil manejar múltiples versiones de los artículos generados así como hacer el seguimiento entre las distintas versiones generadas, los programas utilizados y los datos procesados.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Como se puede apreciar el análisis del conocimiento generado está íntimamente asociado tanto al desarrollo del software de investigación como a la finalización del proyecto mismo. Cuando se genera el conocimiento suficiente no es necesario generar nuevas aplicaciones y se puede finalizar el proyecto.

2.1 Descripción del Problema

Introducción. El contexto de este artículo está dado por lo que se ha caracterizado como "Desarrollo de Software de Investigación". En las secciones anteriores se presentó en detalle en qué consiste este tipo de desarrollos, sus características y dificultades. En esta sección se va a destacar la importancia del tema en estudio y se va a enunciar el problema a resolver de manera clara y precisa.

Enunciado del problema. El problema consiste en **cómo aumentar la productividad de los equipos de desarrollo de software de investigación**. El desafío consiste en definir un proceso de desarrollo que sea fácilmente utilizado por los investigadores y que les permita mejorar su productividad. Esto se debe lograr sin obligarlos a capacitarse en Ingeniería de Software, poniendo especial énfasis en la coordinación de los integrantes del equipo. **Esta mayor productividad dice relación también con el análisis del conocimiento descubierto en el proyecto**. Ambos temas se conjugan en la decisión de terminar el proyecto una vez que los experimentos han permitido descubrir la cantidad de conocimiento suficiente.
Estos dos temas deben ser abordados para dar una solución integral a la situación problema.

Entonces el problema se puede plantear de dos puntos de vista:

- Por una parte se puede analizar el descubrimiento del conocimiento producto del avance de la investigación en sí y de la realización de los experimentos con la ejecución del software.

- Por otra parte se puede analizar el proceso de desarrollo del software de investigación en sí, incluyendo la interacción y la productividad del equipo de desarrollo.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Gráficamente lo planteado se puede resumir en la siguiente figura:

Figura 1: Contexto del problema a solucionar. Fuente: elaboración propia.

Aporte del diálogo a la situación descrita. Habiendo aclarado el contexto del problema es posible analizar cómo se pueden combinar los dos temas en estudio: el aumento de la productividad del equipo de desarrollo y el análisis del conocimiento descubierto en la investigación.

Respecto a lo anterior cabe preguntarse ¿existe una forma de que un equipo de desarrollo aumente su productividad en la construcción de software?

Como dijo Frederick P. Brooks Jr. (1986) en su clásico artículo "No hay balas de plata", una de las propiedades inherentes de la esencia irreductible de una entidad de software es su complejidad. De esta complejidad derivan muchos problemas clásicos del desarrollo de software. Uno de ellos es la "*dificultad de comunicación entre los miembros de un equipo de desarrollo, lo que lleva a productos defectuosos, costos sobrepasados, atrasos en los itinerarios*". Se agrega

en los proyectos de investigación, una nueva capa de comunicación que es necesaria en el equipo y que se refiere a cómo orientar el trabajo a partir de los conocimientos obtenidos con los experimentos. Aparece entonces el problema de la comunicación humana al nivel de un equipo que planifica su trabajo futuro para converger a un estado de conocimiento suficiente.

La investigación de varios años en el Departamento de Ingeniería Informática de la Usach permite proponer el uso del diálogo ya que este es (Leiva-Lobos, Antillanca & Ponce, 2008): "*un proceso guiado por reglas que produce, examina y hace circular significados y cuyo resultado es el dominio lingüístico que posibilita dicha dinámica*".

El trabajo (Leiva-Lobos et al., 2008) propone un marco para el diseño de artefactos del diálogo. Se da mucha importancia al aporte del Diálogo de Bohm (Bohm, 1997) ya que este destaca la primacía de la cooperación sobre la competencia, justamente lo que se requiere para mejorar la productividad y permitir el análisis del conocimiento generado en la investigación. Sin duda que los significados producidos, examinados y que se hacen circular facilitan todo lo anterior.

En el trabajo ya citado El trabajo (Leiva-Lobos et al., 2008) se menciona que existen reglas a las que se debiera someter el diálogo, así como diferentes movidas.

Las reglas, tomadas de Bohm (Bohm, 1997) y citadas por Senge (Senge, 2009) son las siguientes:

- Todos los participantes deben "suspender" sus supuestos (también llamados juicios).
- Todos los participantes deben tener la visión de los demás como colegas (es decir que se debe respetar al otro y a sus opiniones).
- Debe existir un "árbitro" que mantenga el contexto del diálogo. Para esto se define un participante particular llamado facilitador quien se encarga de lograr que el diálogo fluya y que todos participen.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Se plantea la suspensión de los supuestos como un medio para el aprendizaje en equipo (Senge, 2009). Al aplicar esta regla los participantes pueden ingresar en un pensamiento conjunto, de modo que todos los elementos que definen la estructura del mismo queden expuestos. Esto permite que los pensamientos puedan ser examinados en forma personal y colectiva.

Respecto a las movidas cabe destacar que estas son 4, según lo plantea Kantor (Ugalde, Persen & Cerda, 2012):

- Mover - Oponer - Observar - Seguir.

La primera acción siempre debe ser mover.

Gráficamente se pueden ver de la siguiente manera:

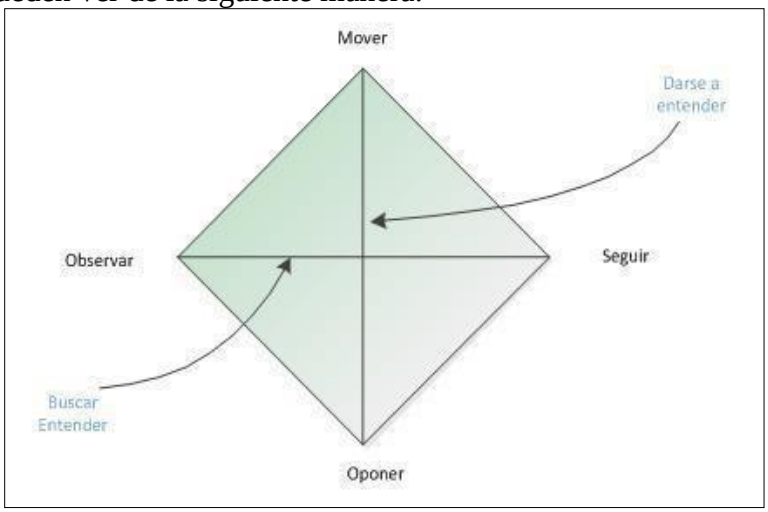

Figura 2: Movidas del Diálogo. Fuente: Leiva-Lobos et al., 2008.

Se espera que la incorporación del Diálogo de Bohm permita mejorar la productividad del equipo de desarrollo tanto en la construcción del software de investigación como en el análisis

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

del conocimiento descubierto. Esta mayor productividad debería estar dada por el diálogo como generador de significados y también por la discusión experta respecto al avance logrado (Senge, 2009).

3. Marco Teórico

3.1 Características de los desarrolladores de software de investigación

Respecto a los desarrolladores se puede decir que (Ko et al. 2011; Segal 2009b):

- Si bien en general han sido escogidos por ser buenos programadores normalmente no tienen las costumbres que propone la Ingeniería de Software en cuanto a seguir un método específico de desarrollo. Este tipo de proyecto tiene similitud con el "Desarrollo exploratorio" (Sommerville, 2011) donde se van generando versiones de la aplicación cada vez más cercanas a lo que se requiere.

- Poseen una buena disposición para incorporar mejoras en el proceso de desarrollo de software siempre que sean fáciles y rápidas de usar.
- Están enfocados en la generación del conocimiento más que en aplicar técnicas de Ingeniería de Software.

En la siguiente tabla se presenta un resumen de las características de este tipo de proyectos.

Tabla 1.

Características de los Proyectos de Software de Investigación.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Atributo	Valores	Descripción
	(a) Individual	Este escenario, a veces llamado del "investigador solitario" implica un solo desarrollador.
(1) Tamaño del equipo de trabajo	(b) Hasta 7 integrantes	Se refiere a un equipo de entre 2 y 7 integrantes que pueden o no estar distribuidos geográficamente
	(c) Grandes equipos	Se refiere a más de 7 desarrolladores, posiblemente distribuidos geográficamente
(2) Vida del código	(a) Breve (hasta un año)	Un código que se ejecuta pocas veces y que posteriormente pierde importancia generado
	(b) Larga (más de un año)	Un código que se ejecuta muchas veces durante su vida útil
(3) Usuarios	(a) Internos	Solo los desarrolladores utilizan el software
	(b) Externos	El código es utilizado por otros grupos en la organización
	(c) Ambos	Los códigos son usados tanto interna como externamente. La generación de las versiones es más compleja y exige controles de calidad más exhaustivos
(4) Calidad del código	(a) Refactorizado	El código es revisado para que su ejecución sea eficiente
	(b) No refactorizado	No es obligatorio optimizarlo para que utilice pocos recursos en el sistema donde se ejecuta. Basta con que funcione.
(5) Foco del	(a) Crear software	Son proyectos que tienen como principal objetivo la proyecto construcción del software.
	(b) No crear software	El principal objetivo es probar una teoría, apoyar una investigación o realizar

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

		un experimento. La creación del software es algo secundario.
(6) Estructura equipo de	(a) Solo Desarrolladores Científicos[3]	Son equipos de proyecto donde el software es desarrollado solamente por científicos. Típicamente trabajo no poseen conocimientos de Ingeniería de Software.
	(b) Desarrolladores Científicos con Informáticos	El desarrollo del software es liderado por Desarrolladores Científicos con el apoyo de Informáticos Científicos quienes poseen una formación en el desarrollo de software pero no en Ingeniería de Software.
	(c) Solo desarrolladores Informáticos Científicos	El desarrollo del software es realizado solamente por Informáticos Científicos.
	(d) Desarrolladores Informáticos Científicos con Ingenieros de Software	El desarrollo del software liderado por Informáticos Científicos con el apoyo de ingenieros informáticos quienes poseen una formación en Ingeniería de Software[5]
(7) Duración del proyecto	(a) Breve (hasta 3 meses)	Se refiere a proyectos con una duración igual o menor a 3 meses
	(b) Larga (más de 3 meses)	Se refiere a proyectos con una duración de más de 3 meses
(8) Respecto al experimento	(a) El uso del software es el experimento	Se refiere a proyectos donde ejecutar y probar un software específico es el objetivo del experimento
	(b) El procesamiento de los datos es el experimento	Se refiere a proyectos donde procesar un conjunto específico de datos es el objetivo del experimento
	(c) Probar los parámetros del software de un experimento	Se refiere a proyectos donde modificar los parámetros de un software y luego probar los resultados es el objetivo del experimento

[3] Se entiende por Desarrollador Científico un investigador que no es informático pero que sabe programar.
[4] Se entiende por Informático Científico un profesional que se especializó en las Ciencias de la Computación. Sin ser un Ingeniero de Software está capacitado para programar.
[5] Se entiende por Ingeniero de Software un profesional informático experto en métodos de desarrollo de aplicaciones informáticas.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Nota: Fuente: Morris & Segal 2012; Segal 2009c.

3.2 Dificultades en el desarrollo y uso del software de investigación

Uno de las principales dificultades que tienen los investigadores es reproducir los experimentos realizados. Dicha dificultad aumenta en proporción al tiempo transcurrido debido al olvido paulatino de los detalles del proyecto. Generalmente no se registra qué versión de la aplicación generó los resultados publicados ni los datos utilizados.

Es difícil planificar este tipo de proyectos y definir sus fechas de término y de entrega de avances ya que al ser un proyecto de investigación no se tiene claro hasta donde se llegará y en cuanto tiempo. A veces se asignan recursos insuficientes a proyectos de cierta complejidad. Lo mencionado dificulta la planificación en la construcción del software de investigación y en el cumplimiento de los plazos de entrega.

Resulta trabajoso modificar las aplicaciones ya construidas debido a que la documentación de las mismas es más bien escasa (Segal, 2005). En general solo se documenta en el mismo código escrito pero sin tener un estándar definido y aceptado para eso. En consecuencia depende de quien haga el código la calidad de la documentación generada. Por las mismas razones es complejo modificar una aplicación desarrollada por otro equipo de proyecto.

En general los errores que se cometen en la gestión de algunos proyectos se repiten en otros ya que se documenta escasamente lo realizado, impidiendo de esta forma crear una base de conocimiento común.

Para estos desarrolladores resulta complejo aplicar herramientas de Ingeniería de Software para solucionar las dificultades descritas. Estos desarrolladores de software se han clasificado en:

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

- Desarrolladores científicos quienes pueden ser de cualquier disciplina. Comparten eso sí la capacidad y el interés de construir aplicaciones.

- Informáticos científicos quienes vienen de las ciencias de la computación manejando diferentes técnicas de programación.

Además se identificó que los desarrollos de software se realizan en un contexto de gran complejidad conceptual por lo que se dedican pocos esfuerzos a la formalización de la construcción de las aplicaciones. El resultado de esto son piezas de software mal documentadas que son difíciles de modificar. Además de esto la ejecución de dichas aplicaciones (actividad denominada experimento) de manera controlada y repetitiva implica grandes dificultades.

Por otra parte resulta difícil evaluar el conocimiento generado y tomar la decisión de construir o no otra versión del software de investigación.

4. Discusión Bibliográfica

4.1. Introducción

Varios de los autores consultados en el estudio bibliográfico de este artículo han llegado a conclusiones parecidas (Segal 2005; Segal 2012): cuando se desarrolla software de investigación existen dificultades para articular los requerimientos y para generar la documentación. Lo primero se debe a la complejidad del dominio del problema y lo segundo se debe a la reticencia de los equipos de desarrollo de aplicar técnicas de la Ingeniería de Software.

Sin embargo se agregan otras dificultades que son propias del desarrollo del software de investigación (Segal, 2005):

- Dificultad para decidir si ya se generó el suficiente conocimiento con el proyecto o es necesaria seguir profundizando.

- Dificultad en la incorporación de un nuevo integrante al proyecto debido a la complejidad del dominio del problema a resolver.

- Dificultad en la repetición de los experimentos con el software desarrollado.

- Dificultades para validar las aplicaciones desarrolladas.

4.2 Aportes al desarrollo de software de investigación

Se ha caracterizado a los desarrolladores de software de investigación como "usuarios finales desarrolladores", generalmente científicos (Segal, 2009d). Esto significa que las mismas personas que construyen las aplicaciones son las que las utilizan y por lo tanto están presionados a tenerlas disponibles lo antes posible. Lo anterior genera errores tanto en la construcción como en la documentación de las aplicaciones, tal como ya se describió.

Además estos desarrolladores son reacios al uso de métodos propios de la Ingeniería de Software ya que prefieren enfocar sus esfuerzos en tener lo antes posible la aplicación funcionando (Ko et al., 2011). Todo lo explicado permite suponer que basarse en las propuestas de los Métodos Ágiles parece una buena solución toda vez que estos se enfocan en la generación de código funcionando como primera prioridad (Morris & Segal 2012; Robinson, Segal & Sharp 2007). La palabra ágil quiere decir que son metodologías que se adaptan muy bien a los cambios en el entorno de desarrollo.

Ha habido también estudios de análisis del método ágil XP (Extreme Programming) (Robinson et al., 2007), uno de los más utilizados en el desarrollo de software. El análisis fue hecho a

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

través del estudio de cuatro equipos de desarrollo muy maduros que usan esa forma de trabajo. Se concluyó que resulta útil incorporar las siguientes prácticas al desarrollo de software:

- La codificación y prueba de lo escrito en pareja

- Una rápida retroalimentación de los avances mediante reuniones

- Generar rápidamente código que funcione y realice algunas de las funciones que se necesitan

- Tener una buena comunicación entre los integrantes del equipo

Todos estos elementos son mencionados como aportes para mejorar la productividad de los equipos de trabajo al desarrollar el software de investigación.

Se destaca también la necesidad de enseñar algunas técnicas mínimas a estos usuarios finales desarrolladores (Umarji, Pohl, Seaman, Koru & Hongfang, 2008). Entre estas cabe destacar: una fuerte formación en control de calidad del software, capacidad para generar documentación fácilmente entendible y la capacidad de reutilizar aplicaciones ya construidas. Incluso se presenta una experiencia práctica donde los usuarios finales fueron capaces de evaluar la aplicación DDtrac construida mediante una metodología de investigación-acción para desarrollar dicha aplicación basada en Web (Dawn, 2009). En esa misma línea se plantea darle más atención a los usuarios de las aplicaciones Web quienes tienen numerosos aportes que hacer ya que cada día muchas de estas personas deben personalizar sus aplicaciones (Macías & Castells, 2007).

4.3 Experiencias logradas desarrollando software de investigación

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Dentro de las actividades identificadas para la construcción de software por parte de usuarios finales desarrolladores cabe destacar (Tan & Phillips, 2005):

- Compartir diferentes roles durante el desarrollo del proyecto. Estos roles pueden ser: analista de sistemas, diseñadores de sistemas, "tester" de sistemas, administrador de proyecto y líderes de equipo.
- Mantener actualizado un tablero de proyecto con la información más relevante del mismo.

- Realizar reuniones semanales de coordinación.

- Liberar documentos periódicos relativos a los avances del proyecto.

Una de las conclusiones de la experiencia descrita (Tan & Phillips, 2005) es la importancia de las habilidades "blandas" o "sociales" que dicen relación con la comunicación y el trabajo en equipo. Respecto a lo mismo otros autores (Howcroft & Wilson, 2002) se refieren a las contradicciones que se pueden ver en los usuarios que participan en proyectos de sistemas de información poniendo énfasis en la naturaleza conflictiva de las relaciones organizacionales así como un conjunto de paradojas que este fenómeno genera. Para ejemplificar se presenta la analogía del dios romano de dos cabezas, Jano en relación con el papel del desarrollador de sistemas. Esta analogía enfatiza la incompatibilidad de algunas de las necesidades de los miembros de la organización. Resolver estas situaciones requiere de grandes dosis de habilidades sociales algo que se destaca también en el estudio del desarrollo de una aplicación basada en el "registro centrado en el paciente" (Puentesa, Roux, Montagner & Lecormu, 2012).

Así como se menciona el rol de "tester" (Tan & Phillips, 2005) que se debe integrar al proyecto también se investiga la coordinación entre los desarrolladores y los "testers" en la construcción de software (Dhaliwal, Onita Poston & Zhang, 2011). Con esto se busca entender la coordinación dentro de la unidad de TI. La evidencia de tensiones dentro de estas unidades da

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

una pista de que existen conflictos no resueltos. Los resultados de la investigación sugieren que la dimensión de las relaciones es más relevante que la estructura de las unidades de TI. Se podría concluir entonces la necesidad de considerar como una variable relevante el buen manejo de las relaciones entre los integrantes del equipo de desarrollo.

Por otra parte el dominio del problema con que se trabaja en la investigación influye mucho en el éxito del proyecto. Puede requerirse mucho tiempo para capacitar a los participantes en los temas más relevantes con los que deben trabajar (Mollá & Castells, 2006). Por este motivo se han aplicado diferentes técnicas entre las que cabe destacar el uso del "Modelo de Conocimiento Científico" (Li, 2011) que se resume en la siguiente figura:

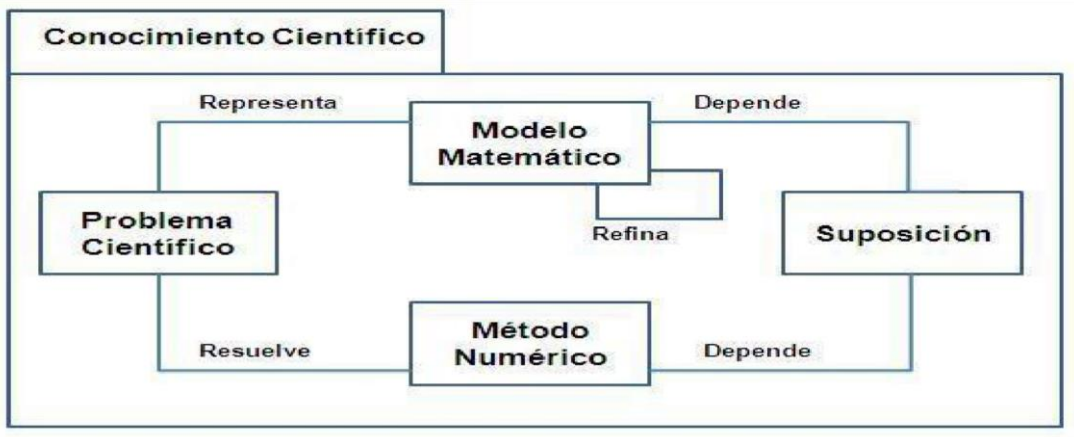

Figura 3: Modelo de Conocimiento Científico. Fuente: Li, 2011.

Lo planteado se resume en definir primero el problema científico. Luego crear el modelo matemático que lo describe e identificar los factores que influyen en el problema científico. Por último aplicar los métodos numéricos para resolver el problema ya definido. En algunos casos el dominio del problema estudiado puede estar muy cercano a la electrónica, como cuando se analiza una arquitectura de hardware para sacar el máximo provecho procesando grandes volúmenes de datos (Kurzak, Tomov & Dongarra 2012). Sin embargo los modelos matemáticos son fundamentales para resolver este tipo de problemática.

La mala documentación del código construido también atenta con el éxito de los desarrollos. Es por esta razón de que se propone generar un buen diseño haciendo ingeniería reversa a partir del código construido que solucionó el problema (Li, 2011).

También resultan interesantes las experiencias acumuladas en el desarrollo de software para la disciplina química (Peachey et al. 2013) ya que a partir de lo aquí aprendido se ha generado el concepto de e-Science (Spencer, Zimmerman & Abramson, 2011) que consiste en la colaboración de distintos equipos de trabajo distribuidos en diferentes partes del mundo. Este desafío aumenta la complejidad de administrar los equipos de trabajo al agregarse a lo ya analizado la presión de comunicarse con diferentes husos horarios, idiomas y culturas. Sin embargo en el desarrollo de software de investigación es inevitable la interacción de investigadores y científicos repartidos por el mundo, sobre todo cuando se trabaja en temas avanzados ya que la cantidad de personas que los conocen tiende a ser escasa. Nuevamente se destaca la importancia de una buena coordinación entre los integrantes del equipo.

Además de la utilidad de coordinar equipos de trabajo por todo el mundo tal como lo presenta el concepto de e-Science existe también la necesidad de coordinar la capacidad de cómputo también entre centros de investigación distribuidos entre diferentes países. Como ejemplo de esto se pueden citar los conceptos de "Many-Task Computing" (Abramson, Bethwaite, Enticott, Garic & Peachey, 2011), "Thermo Data Engine" (TDE) (Diky et al., 2011) y "tile algorithms" (Ltaief, Kurzak & Dongarra, 2010).

Por otra parte también se desarrolla el tema del aporte de los proyectos de código abierto a la computación de alto rendimiento, tema crucial para las investigaciones científicas (Dongarra et al., 2009). Se plantea que si bien estos proyectos han sido un avance para el desarrollo de este tipo de aplicaciones faltan mecanismos que permitan detectar los errores del código más rápidamente y corregirlos de manera oportuna. También se comenta que si bien un repositorio común y una lista de e-mails para realizar discusiones pueden ser suficientes para coordinar un proyecto individual, no lo son para que la comunidad internacional, dedicada a la

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

computación de alto rendimiento, pueda hacer sus aportes de manera eficiente (Dongarra et al., 2009). Este planteamiento da un buen ejemplo de lo complejo que resulta coordinar equipos de desarrollo. La dificultad está dada por la necesidad de coordinar muchos computadores en pos de la computación a escala peta (mil billones de equipos o 1015) o exa

(un trillón de equipos o 1018) tal como lo requiere el proyecto "Blue

Waters" de la National Science Foundation (NSF). Dentro de las propuestas para superar estas dificultades se encuentran:

- La creación de un marco de trabajo (framework) para organizar el desarrollo de software de investigación comunitaria.

- Realizar un análisis detallado de las necesidades, problemas y estrategias a utilizar.

- Crear una hoja de ruta de la coordinación del software.
- Fomentar y facilitar la colaboración en materia de educación y formación.

- Coordinar a la comunidad de proveedores en un eje transversal de esfuerzos.

Hay autores que analizan el problema de lograr una buena interacción entre la ciencia y la tecnología, enfocándose en la recolección de los datos que se requieren procesar (Borgman, Wallis & Mayernick, 2012). Para esto presentan un estudio de colaboración a largo plazo entre científicos del medio ambiente (biología, ecología, ciencias del mar), científicos informáticos y equipos de investigación en cinco universidades y centros tecnológicos de investigación. Este estudio está hecho a partir de cuatro preguntas de investigación:

1. ¿Cuáles son los "datos" en investigaciones donde colaboran la ciencia y la tecnología?

2. ¿Cómo varía el concepto de "dato" según el objetivo de la actividad de investigación?

3. ¿Qué rol desempeñan los datos en las colaboraciones entre la ciencia y la tecnología?

4. ¿Qué se puede aprender sobre el trabajo colaborativo siguiendo los datos usados?

Dentro de sus conclusiones cabe destacar que cuando los equipos de trabajo crecen en tamaño y la tecnología incrementa su complejidad las interdependencias se multiplican. Ellos constataron también que los investigadores de ciencia y tecnología dependen de los datos de otros para poder interpretar los que les son propios. Muestran que en muchos casos los datos que se procesan en la investigación no requieren ser administrados en forma conjunta sino de manera separada. Además este tipo de investigación no tiende a generar datos que sean útiles para investigadores que sean externos al proyecto. Finalmente destacan que la colaboración exitosa depende de que cada equipo de trabajo trabaje de buena fe tratando de capturar los mejores datos posibles según los estándares definidos con sus socios de investigación.

En la línea de lograr equipos de trabajo de alto rendimiento (Peter, 2010) propone ocupar la visualización de aplicaciones, técnica usada por profesionales de desarrollo de software. Para esto presenta una visión de cuatro estudios empíricos: (1) imaginación mental de programadores expertos, (2) cómo estos expertos comparten su imaginación con el resto del equipo de trabajo, (3) cómo utilizan estos expertos las aplicaciones visuales estándares y (4) qué herramientas construyen estos expertos para este fin y cómo las usan. Por su parte otros autores sugieren preparar a las personas que participarán en los proyectos de desarrollo de software de investigación para que sean capaces de combinar el modelamiento matemático, la Ingeniería de Software y el diseño centrado en el usuario, tal como ellos lo aplican en la

"University of Southern Denmark" (Angelov, Melnik & Buur, 2003). Por último (Dall'Osso, 2003) propone el uso de componentes reutilizables ya que es una opción que puede ser explotada por los usuarios finales desarrolladores.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

4.4 Respecto a la productividad en el desarrollo de software

Tal como se mencionó anteriormente uno de los desafíos cuando se enfrenta el desarrollo de software de investigación es la necesidad de aumentar la productividad. Si bien esta situación no es exclusiva de este tipo de software ya que es una de los desafíos que se han presentado desde que se empezó la construcción de programas computacionales la situación agrega nuevas dificultades. Estas dificultades dicen relación con el hecho de que los requerimientos del software de investigación a construir se extraen del dominio de la situación investigada no existiendo el concepto de "Cliente", tan común en el desarrollo de software tradicional (Sommerville, 2011). Por esta razón es difícil predecir la productividad del equipo de desarrollo ya que además de la complejidad natural de la construcción del software se agregan otros factores tales como: la complejidad del dominio del tema investigado, la imposibilidad de identificar los requerimientos al inicio del proyecto por el poco conocimiento generado y por último la posibilidad de que se generen subproyectos no previstos inicialmente.

El concepto de productividad se ha enfocado desde dos puntos de vista (Sommerville, 2011):

1. Usando medidas relacionadas con el tamaño, por ejemplo la cantidad de líneas de código.

2. Usando medidas asociadas a la funcionalidad de la aplicación construida.

Sin embargo el tamaño en general y la cantidad de líneas de código no son relevantes en el contexto del desarrollo de software de investigación, ya que a los desarrolladores no se les recompensa por este concepto ni tampoco se les presiona para que escriban más o menos de ellas. Por otra parte la funcionalidad de la aplicación si es relevante ya que es el foco del desarrollo al estar asociado al concepto de experimento que se explicó previamente.

Debido a lo anterior la productividad será considerada desde el punto de vista de cómo lograr la funcionalidad requerida en el menor tiempo posible. Al respecto James Howison y Herbsleb (2011) dan algunas pistas al mencionar que se debe haber una relación entre los incentivos de

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

los participantes en el proyecto de investigación y la necesidad de crear el software. Esta relación de incentivos, platean ellos, permite mejorar la colaboración. Por su parte (Basili, Carver, Nochstein, Hollingsworth, Zelkowitz & Shull, 2008) destacan que a los científicos los miden por la publicación de papers y no por la construcción de software.

La conclusión de este análisis es que el aumento de la productividad debe estar medido por la creación del software con la calidad suficiente pero en menos tiempo.

5. Presentación de KDDP como propuesta de solución

Una vez analizada la situación problema es posible presentar la propuesta de solución: el Método KDDP (Knowledge Discovery-Driven Project, es decir Proyectos Guiados por el Descubrimiento de Conocimiento). Esta propuesta toma elementos de los métodos ágiles, utiliza el Modelo de la Espiral (Boehm, 1988) para apoyar la aproximación paulatina al conocimiento que se busca generar con la investigación. También utiliza el Diálogo de Bohm (Bohm, 1997) para hacer circular los significados entre los participantes de la investigación.

A continuación se describe en detalle el método.

Las herramientas que utiliza KDDP y los roles que define son los siguientes (Antillanca, 2011):

Herramientas: se definen las siguientes herramientas de apoyo:
- Reuniones diarias de coordinación de 15 minutos "de pie". Cada integrante responde las preguntas: ¿qué hice ayer?, ¿qué voy a hacer hoy? y ¿qué dificultades he tenido?

- Una Wiki del proyecto para los compromisos establecidos.

- Para la documentación se sugieren las siguientes aplicaciones:

- El generador automático de documentación Doxygen para el código.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

- El sistema de control de versiones SVN (Subversión) para el software y la documentación.

- Una "Matriz de Trazabilidad" donde quede registrado cada versión de software generado, los datos que procesó, los resultados que obtuvo, en qué fecha, en que computador se encuentra cada versión y que Proyecto fue el responsable de la creación de la aplicación. A modo de ejemplo se presenta la siguiente figura:

Proyecto: Proyecto 002 – 2009		**Equipo de Trabajo:** Equipo 1	
Ciclo	**Aplicación creada**	**Desarrollador:**	**Datos ejecución:**
1	Pr005	ABC, HIJ …	Pr005.dat
…			
N			

…

Inicio: 20/12/2009	
Fecha ejecución:	**Documento de resultados:**
01/01/2010	Inf005.doc

Figura 4: Matriz de Trazabilidad

Donde se muestra que se registra el nombre del proyecto, el nombre del equipo que realizó el proyecto y la fecha de inicio del mismo.

Para cada ciclo se debe registrar:

- La aplicación creada identificada con un nombre único.

- El o los desarrolladores responsables.
- Los datos de ejecución utilizados, identificados de forma única.

- La fecha de ejecución de la aplicación con esos datos.

- Los documentos de registro y análisis resultantes.

Cada equipo de trabajo define la ubicación de sus aplicaciones, datos de ejecución y documentos de resultados. Estas definiciones son conocidas por todos los equipos.

Roles: en cada equipo de trabajo existen los siguientes roles y responsabilidades:

- Jefe de Proyecto: orientar y liderar el trabajo del equipo. Además asume el rol de Facilitador cuando se desarrolla el diálogo.

- Desarrollador-Documentador: escribir el código, documentar la programación.

- Encargado de Datos y Tester: preparar los datos a procesar, ejecutar los programas creados, revisar los resultados.

Cabe mencionar que es posible definir otros roles que no sean permanentes durante el desarrollo del proyecto.

Paso a paso: para complementar la presentación de KDDP se incluye un "paso a paso" del método (Ugalde et al., 2012):

Figura 5: Modelo propuesto en espiral. Fuente: adaptado de Sommerville, 2011.

La explicación de cada cuadrante indica que se debe realizar.

Cuadrante 1 (C1):

Definición del objetivo para el ciclo. Esta actividad está liderada por el Jefe de Proyecto pero cuenta con la participación y aporte del Equipo de Trabajo completo. La duración del ciclo debería ser entre 1 y 2 semanas. En el Cuadrante 2 se define la duración del ciclo.

Asignación de tareas para los integrantes del Equipo de Trabajo.

Resultados:

Un breve informe con el objetivo a lograr en el ciclo descrito en una Minuta de Reunión.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Cada integrante tiene su actividad definida (uso de la Wiki).

Cuadrante 2 (C2):

Investigación previa: cada integrante realiza una breve investigación (no más de 2 días) para cuantificar los tiempos que requerirá para completar su tarea.

Respecto a las actividades definidas:

Se evalúan las diferentes alternativas para realizar las tareas asignadas.

Se escoge una de las alternativas posibles.

Si el ciclo implica construcción de software se definen los requerimientos que deberá cumplir la aplicación que se desarrollará. El Equipo de Trabajo define de común acuerdo el plazo de construcción de la aplicación y los roles específicos que cumplirá cada uno: DesarrolladorDocumentador, Encargado de Datos y Tester (uso de la Wiki). Puede haber más de un integrante en cada rol.

Resultados:

Cada integrante del Equipo tiene asignada una actividad a realizar con un plazo de entrega.

Si el ciclo implica construcción de software:

Una lista de funcionalidades y características que la aplicación deberá cumplir.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Una asignación de los roles para cada integrante.

Cuadrantes 2 y 3 (C2-3):
Reunión de análisis: el Equipo de Trabajo analiza los pasos a seguir acordando las acciones para la siguiente semana. Las acciones se definen indicando: responsable y plazo para completarla (uso de la Wiki).

Si el ciclo implica construcción de software: se crea el código de la aplicación (uso de Doxygen y SVN). Cada integrante reporta diariamente sus avances, dificultades y descubrimientos (reunión de coordinación de 15 minutos "de pie").

6. Resultados

Una lista actividades asignadas.

Eventualmente una lista de descubrimientos realizados por los integrantes del equipo durante el análisis (actividad C2-3).

Cuadrante 3 (C3):

Cada integrante del Equipo realiza la actividad asignada.

Si el ciclo implica construcción de software:

Se crea el diseño, se definen las pruebas a realizar, se codifica y se realiza el experimento con los datos seleccionados ("Matriz de Trazabilidad"). Se analizan los resultados.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Se repite el paso hasta que el grupo esté satisfecho con los resultados obtenidos (uso de Doxygen y SVN).

Resultado: un informe de análisis con los resultados obtenidos.

Cuadrante 4 (C4):

En este cuadrante se realizará el diálogo y se decidirá si se realiza o no otra vuelta de la espiral. Ocupando las movidas del diálogo ya descritas será posible analizar el conocimiento descubierto concluyendo de esta manera si es necesario o no realizar otro ciclo. Este cuadrante es el que permite llamar a KDDP Knowledge Discovery-Driven, es decir guiado por el descubrimiento de conocimiento.

6. Discusión y conclusión

Después de analizar la documentación seleccionada se pueden extraer las siguientes conclusiones:

- El desarrollo de software de investigación está en pleno crecimiento en los últimos años. Todo indica que cada vez va a tomar una mayor relevancia.
- Existen múltiples interrogantes respecto a cómo gestionar este tipo de proyectos que aún no tienen respuesta. Entre estas cabe destacar la necesidad de coordinar a los equipos de desarrollo de software y también en cómo evaluar el conocimiento generado por la investigación.

- La Ingeniería de Software ha hecho aportes a este tipo de desarrollo pero las técnicas que propone no son adoptadas por la mayoría de los desarrolladores al no ser estos profesionales informáticos.

- Se están buscando herramientas que faciliten el trabajo de los equipos de investigación y aumenten su productividad. Dentro de estas herramientas se destacan algunas muy técnicas tal cómo la reutilización de código pero también otras más cercanas a las habilidades sociales.

- Se estima que los métodos ágiles pueden proporcionar formas de trabajo y coordinación que impacten positivamente en estos tipos de desarrollo de software.

- En general las universidades no han definido un perfil específico para un desarrollador científico que lo prepare para liderar este tipo de proyectos.

Por lo tanto se asume que el Método KDDP permitirá una interacción positiva entre Investigadores, Informáticos Científicos e Ingenieros de Software para aumentar los resultados de los proyectos donde se desarrolle software.
En un artículo posterior se describirán los resultados obtenidos con la aplicación de KDDP.

7. Referencias

Abramson, David; Bethwaite, Blair; Enticott, Colin; Garic, Slavisa & Peachey, Tom (2011). Parameter Exploration in Science and Engineering Using Many-Task Computing.

IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 6, June.

Angelov, C.; Melnik, R. V. N. & Buur J. (2004). The synergistic integration of mathematics, software engineering, and user-centred design: exploring new trends in education.

Future Generation Computer Systems, Volume 19, Issue 8, December

Antillanca Espina, Héctor & Cerda Neumann, Gerardo (2012). Desarrollo de Software Científico. IWASE' 2012, Jornadas Chilenas de Computación, Valparaíso, Chile, Noviembre (paper)

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Antillanca Espina, Héctor & Cerda Neumann, Gerardo (2011). Aplicación de Método para el Desarrollo de Software de Investigación. Jornadas Chilenas de Computación, Curicó, Chile, Noviembre (paper)

Antillanca Espina, Héctor & Cerda Neumann, Gerardo (2010). Propuesta Metodológica para el Desarrollo de Software de Investigación. INFONOR, Antofagasta, Chile, Noviembre (paper)

Arriagada Mena, Marcos & Cerda Neumann, Gerardo (2012). Desarrollo de herramienta CASE para el uso de Metodología KDDP. Santiago de Chile: Universidad UCINF, Trabajo de Titulación presentado en conformidad a los requisitos para obtener el Título de Ingeniero de Ejecución en Informática.

Basili, Victor R.; Cruzas, Daniela; Carver, Jeffrey C.; Hochstein, Lorin M.; Hollingsworth, Jeffrey K.; Zelkowitz, Marvin V. & Shull, Forrest (2008). Understanding the High-Performance-Computing Community: A Software Engineer's Perspective, IEEE Software, July/August.

Boehm, Barry W. (1988), A Spiral Model of Software Development and Enhancement. Computer, May.

Bohm, David, Sobre el diálogo (1997). Barcelona, España: Kairós.

Borgman, Christine L.; Wallis, Jillian C. & Mayernik, Matthew S. (2012). Who's Got the Data? Interdependencies in Science and Technology Collaborations, Computer Supported Cooperative Work, 21

Brooks, Frederick P. Jr. (1986), No silver bullets – Essence and Accident in Software Engineering, Proceedings of the IFIP Tenth World Computing Conference.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Canales, Cerón, Coordinador – editor, Metodologías de investigación social: introducción a los oficios (2006). Santiago, Chile: LOM.

Dall'Osso, Aldo (2003). Using computer algebra systems in the development of scientific computer codes, Future Generation Computer Systems, 19

Dawn, Gregg (2009). Developing a collective intelligence application for special education, Decision Support Systems, November, Volume 47, Issue 4

De Roure, David (2009). Software Design for Empowering Scientists, IEEE Software, 88 – 95.

Dhaliwal, Jasbir; Onita Colin, Gabriel; Poston, Robin & Zhang, Xihui Paul (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers, The Journal of Strategic Information Systems, December, Volume 20, Issue 4, Pages 323 - 342

Diky, Vladimir; Chirico, Robert D.; Kazakov, Andrei F.; Muzny, Chris D.; Magee, Joseph W.; Abdulagatov, Ilmutdin; Kang, Jeong Won; Kroenlein, Kenneth & Frenkel,

Michael (2010). ThermoData Engine (TDE): Software Implementation of the Dynamic Data Evaluation Concept. Experiment Planning and Product Design, December, J. Chem. Inf. Model, 51

Dongarra, Jack; Beckman, Pete; Aerts, Patrick; Cappello, Frank; Lippert, Thomas; Matsuoka, Satoshi; Messina, Paul; Moore, Terry; Stevens, Rick; Trefethen, Anne & Valero, Mateo (2009). The International Exascale Software Project: a Call To Cooperative Action By the Global High-Performance Community, International Journal of High Performance Computing Applications, October.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Fogli, Daniela & Parasiliti, Provenza Loredana (2011). A meta-design approach to the development of e-government services, Journal of Visual Languages and Computing,

November, 23, Issue 2

Howcroft, Debra & Wilson Melanie (2003). Paradoxes of participatory practices: the Janus role of the systems developer, Original Research Article Information and Organization, January, Volume 13, Issue 1, Pages 1-24

Howison, James & Herbsleb, James D. (2011). Scientific Software Production: Incentives and Collaboration, CSCW, March, Hangzhou, China

Isaacs, W. (1999). Dialogue: The Art Of Thinking Together, Broadway Business, 1 edition.

Kelly, Diane & Smith, Spencer (2010). 3rd CASCON Workshop on Software Engineering for Science, Workshop on Software Engineering for Science

Ko, Andrew J.; Abraham, Robin; Beckwith, Laura; Blackwell, Alan; Burnett, Margaret; Erwig, Martin; Scaffidi, Chris; Lawrance, Joseph; Lieberman, Henry; Myers, Brad; Rosson, Mary Beth; Rothermel, Gregg; Shaw, Mary & Wiedenbeck, Susan (2011).

The State of the Art in End-User Software Engineering, ACM Computing Surveys, April, Vol. 43, No. 3

Kurzak, Jakub; Tomov, Stanimire & Dongarra, Jack (2012). Autotuning GEMM Kernels for the Fermi GPU, IEEE Transactions on Parallel and Distributed Systems, November, Vol. 23, No. 11

Leiva-Lobos, Edmundo; Antillanca, Héctor & Ponce Héctor (2008). Un marco sistémico para orientar el diseño de artefactos del diálogo, Revista Convergencia, año 15, número 47, México, páginas 11 – 37, 2008.

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Li, Yang (2011). Reengineering a Scientific Software and Lessons Learned, ICSE '11, May, Waikiki, Honolulu, HI, US.

Ltaief, Hatem; Kurzak, Jakub & Dongarra, Jack (2010). Parallel Two-Sided Matrix Reduction to Band Bidiagonal Form on Multicore Architectures, IEEE Transactions on Parallel and Distributed Systems, April, Vol. 21, No. 4

Macías, José A. & Castells, Pablo (2007). Providing end-user facilities to simplify ontologydriven web application authoring, Interacting with Computers.

Mollá, Diego & Vicedo, José Luis (2006). Question Answering in Restricted Domains: An Overview, Computational Linguistics, October, Volume 33, Number 1

Morris, Chris & Segal, Judith (2012). Lessons Learned from a Scientific Software Development Project, IEEE Software, July / August

Nguyen-Hoan, Luke; Flint, Shayne & Sankaranarayana, Ramesh (2010). A Survey of Scientific Software Development, ESEM'10, September, Bolzano-Bozen, Italy

Peachey, Tom; Mashkina, Elena; Lee, Chong-Yong; Enticott, Colin; Abramson, David; Bond, Alan M.; Elton, Darrell; Gavaghan, David J.; Stevenson, Gareth P. & Kennedy, Gareth F.(2013). Leveraging e-Science infrastructure for electrochemical research, Philosophical transactions of The Royal Society, January.

Peter, Marian (2010). Mental imagery and software visualization in high-performance software development teams, Journal of Visual Languages y Computing, June, Original Research Article, Volume 21, Issue 3

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Puentesa, John; Roux, Michèle; Montagner, Julien & Lecornu Laurent (2012). Development framework for a patient-centered record, Image and Information Processing, Telecom Bretagne

Robinson, Hugh; Segal, Judith & Sharp, Helen (2007). Ethnographically-informed empirical studies of software practice, Information and Software Technology, June, Volume 49, Issue 6

Segal, Judith (2005). When Software Engineers Met Research Scientists: A Case Study, Empirical Software Engineering, 10, 517 – 536.

Segal, Judith (2007). End-User Software Engineering and Professional End-User Developers, Dagstuhl Seminar Proceedings 07081 End-User Software Engineering, 1 – 2, 2007.

Segal, Judith (2008a). Scientists and software engineers: A tale of two cultures, Proceedings of the Psychology of Programming Interest Group, PPIG 08, 10-12- September, University of Lancaster, UK.

Segal, Judith (2008b). Understanding the High-Performance-Computing Community: A Software Engineer's Perspective, IEEE Software, 29 - 36.

Segal, Judith & Morris, C. (2008c). Developing scientific software, IEEE Software, 25(4) : 18 - 20, July 2008.

Segal, Judith (2009a). Software Development Cultures and Cooperation Problems: A Field Study of the Early Stages of Development of Software for a Scientific Community, Computer Supported Cooperative Work (CSCW)

Cerda Neumann, G. y Antillanca Espina, H. (Julio-Agosto 2017). Presentación del Método KDDP para la creación de Software de Investigación. *Revista Akadèmeia, 16* (1), 131-168.

Segal, Judith (2009b). Software Design for Empowering Scientists, IEEE Software, 88 – 95

Segal, Judith (2009c). Software Development Cultures and Cooperation Problems: A Field Study of the Early Stages of Development of Software for a Scientific Community, Computer Supported Cooperative Work (CSCW), 581–606.

Segal, Judith (2009d). Some challenges facing software engineers developing software for scientists, ICSE 09 Workshop, SECSE 09, 9 -14 May, Vancouver, Canada.

Segal, Judith (2011). Obstacles and opportunities with using visual and domain-specific languages in scientific programming, IEEE Symposium on Visual Languages and Human-Centric Computing, 9 - 16.

Segal, Judith (2012). Lessons Learned from Scientific Software Development Project, IEEE Software: 9 - 12, July/August 2012.

Senge, Peter (2009). La quinta disciplina en la práctica. Argentina: Gránica,

Sommerville, Ian (2011). Software Engineering, ninth edition. USA: Addison Wesley.

Spencer, Dimitrina; Zimmerman, Ann & Abramson, David (2011). Special Theme: Project Management in E-Science: Challenges and Opportunities, Computer Supported Cooperative Work.

Tan, Joo & Phillips, John (2005). Real-World Project Management in the Academic Environment, May, CCSC: Northeastern Conference

Ugalde Peña y Lillo, Sergio; Persen Arrepol, Patricio & Cerda Neumann, Gerardo (2012). Evaluación de la propuesta metodológica KDDP para crear software de investigación. Santiago de Chile: Universidad UCINF, Trabajo de Titulación presentado en

conformidad a los requisitos para obtener el Título de Ingeniero de Ejecución en Informática.

Umarji, Medha; Pohl, Mark; Seaman, Carolyn; Koru, Güneş A. & Hongfang, Liu (2008). Teaching Software Engineering to End-users, WEUSE IV'08, May, 2008, Leipzig,

Germany