Modelo Dinámico de Simulación para la Gestión de Proyectos de Software Desarrollados con XP

Diego Alberto Godoy[1] , Tamara Gisel Kasiak[1]

[1] Universidad Gastón Dachary, Centro de Investigación en Tecnología de la Información y Comunicaciones, Departamento de Ingeniería y Ciencias de la Producción.
{diegodoy,tamarakasiak}@dachary.edu.ar

Resumen. En este trabajo se presenta un Modelo Dinámico de Simulación que puede ser utilizado como una herramienta de ayuda en la gestión de proyectos de desarrollo de software llevados a cabo con la metodología Programación Extrema. De acuerdo con esto, a lo largo del trabajo se hace énfasis en demostrar la utilidad de este simulador en la toma de decisiones frente a situaciones típicas que pueden ocurrir en este tipo de proyectos, como ser el retraso en la entrega de una iteración o versión, el abandono del proyecto por parte de programadores, la incorporación de requerimientos durante el desarrollo, entre otras cuestiones. De esta forma, con la ayuda de esta herramienta los administradores de proyectos podrán observar los resultados de cada decisión tomada y elegir la mejor alternativa para ser aplicada en el proyecto real.

Palabras Claves: Sistemas Dinámicos. Programación Extrema. Proyectos de Software.

1 Introducción

En el ámbito de la Ingeniería de Software, la evolución de las Metodologías de Desarrollo de Software ha llevado a la aparición de las denominadas Metodologías Ágiles, las cuales están destinadas a acabar con la rigidez y burocracia de las Metodologías Tradicionales, caracterizadas por la documentación excesiva del proceso de desarrollo y la inflexibilidad ante los cambios que puedan ocurrir en el mismo.

En este trabajo se ha estudiado específicamente la Programación Extrema (XP) [1], una de las más importantes y reconocidas entre las Metodologías Ágiles. La misma fue desarrollada para que equipos de tamaño mediano a pequeño desarrollen software frente a requisitos imprecisos y muy cambiantes. Cabe mencionar que la palabra "Extrema" en el nombre de este método se debe a que XP toma principios y prácticas ya conocidas en el ámbito del desarrollo de software y los lleva a niveles extremos, es decir, busca ejecutar en paralelo todas las prácticas de desarrollo de software y que las mismas se complementen unas a otras de la mejor manera posible.

De esta forma, de acuerdo con las características de XP presentadas por su autor Kent Beck en [1], esto es sus valores, principios y prácticas, la administración de proyectos de software usando esta metodología es compleja e impredecible debido a la gran cantidad de variables que el administrador debe tener en cuenta.

No obstante, una de las alternativas ante estas circunstancias es la utilización de Modelos Dinámicos de Simulación que permiten evaluar los resultados de diferentes decisiones de gestión, sin interferir en el desarrollo real del proyecto. Sin embargo, la mayoría de estos modelos están destinados a simular proyectos desarrollados con Metodologías Tradicionales, como ser el Modelo de Abdel-Hamid y Madnick [2], el Modelo Dinámico Reducido [3], el Modelo SEPS [4] y el Modelo del Laboratorio Draper [5], entre otros.

Si bien ya se han realizado algunos trabajos con el objetivo de estudiar el comportamiento de Proyectos de Desarrollo de Software con XP, a través del uso de modelos dinámicos y simulación [6, 7], como respuesta a la situación mencionada anteriormente en este trabajo se presenta un "Modelo Dinámico de Simulación para Proyectos de Software llevados a cabo con Programación Extrema", con el objetivo de analizar los efectos del uso de prácticas XP en la gestión de proyectos de desarrollo de software y servir como herramienta de ayuda a los administradores en la toma de decisiones en proyectos XP.

El modelo presentado refleja el uso de prácticas XP y su estructura representa el estilo de desarrollo iterativo e incremental, permitiendo simular una versión a la vez, con sus respectivas iteraciones. Así también, permite que los administradores realicen cambios en variables críticas como ser la cantidad de requerimientos a desarrollar en cada iteración, el tiempo de entrega de las mismas, la cantidad de programadores, las horas de trabajo por día, entre otras. Es decir que el modelo permite responder a preguntas del tipo "¿Qué pasaría si...?". De esta forma, los administradores podrán evaluar el impacto que tendría una o varias decisiones de gestión, compararlas entre sí y escoger la mejor de ellas para ser aplicada al proyecto real, sin comprometer el desarrollo del mismo. Cabe destacar que este trabajo complementa el trabajo previamente presentado en [8].

2 Construcción del Modelo

Para construir el modelo mencionado se ha utilizado el Software VenSim PLE 5.4c (Versión Académica) [9] y se han seguido las etapas de la Metodología de Dinámica de Sistemas [10], la cual consta de tres fases.

De esta forma, en la Fase de Conceptualización se ha construido el Diagrama Causal el cual representa, a través de variables e interrelaciones entre las mismas, la estructura del sistema modelado.

Luego en la segunda fase, denominada Fase de Formulación se construyó el Diagrama de Forrester el cual constituye una traducción de las variables y relaciones del Diagrama Causal a ecuaciones matemáticas, posibles de ser programadas y simuladas. Este Diagrama de Forrester, que se compone de 143 variables, fue además dividido en 12 Subsistemas Conservativos de acuerdo con [11], para facilitar su estudio y análisis. Dichos subsistemas son los que se nombran a continuación: Recursos Humanos, Factor de Carga (FC), Desarrollo de Pruebas de Unidad,

Desarrollo de Pruebas de Aceptación, Presión en el Plazo, Planificación, Desarrollo de Historias de Usuario (HU), Desarrollo de Tareas [8], Desarrollo de Tareas Extras, Horas de Trabajo por Día, Actualización de los Días Ideales de Ingeniería por HU y por Tarea y además un subsistema que agrupa las variables auxiliares y constantes del modelo.

Finalmente, en la tercera fase de la metodología utilizada, la Fase de Evaluación se han realizado corridas experimentales y de validación del modelo, utilizando para ello casos reales y escenarios que representan situaciones frecuentes en proyectos XP. Estas corridas tuvieron como objetivo analizar las diferentes decisiones y escenarios que son posibles evaluar a través de la utilización del modelo.

Cabe mencionar que las prácticas XP presentadas por Kent Beck en [1], que fueron posibles de representar en el modelo, utilizando Dinámica de Sistemas son las siguientes: "El Juego de la Planificación", "Versiones Pequeñas", "Hacer Pruebas", "Re-codificación", "Programación en Parejas", "Integración Continua" y "40 horas semanales".

A continuación en la sección 3 se presentará un caso de validación real del modelo y luego algunos de los experimentos realizados con el mismo.

3 Validación del Modelo

Para la validación del modelo presentado se han utilizado datos de tres proyectos de software reales desarrollados con XP.

El primero y más complejo de ellos fue un "Proyecto de Desarrollo de un Sistema de Gestión de una Empresa de Confecciones" [12], el cual consistió en el desarrollo de una versión, dividida en tres iteraciones. Por otro lado, fueron 16 las historias de usuarios (HU) planificadas para ser desarrolladas en dicha versión, por un equipo de 7 programadores. No obstante, al iniciar el desarrollo el cliente decide quitar 4 de las 16 HU planificadas, restando solamente 12.

El detalle de las iteraciones se puede observar en la Tabla 1.

Tabla 1. Detalle de las iteraciones del proyecto presentado en [12]

Iteración	Historias de Usuario (HU)	Tareas en Promedio por HU	Duración Real en días	Duración Estimada en días
1	2	8	21	7,5
2	3	12	18	9,5
3	7	18	8	19,7
TOTAL	**12**	**38**	**47**	**36,7**

De acuerdo con los datos presentados en la Tabla 1, en la Figura 1 se puede observar la planificación de las HU para cada una de las iteraciones. Esta planificación es representada en el modelo por el "Flujo de planificación de HU".

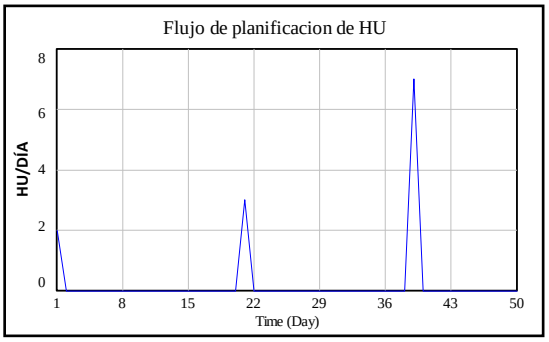

Fig. 1. Planificación de HU para cada iteración

Cabe mencionar además que la planificación de iteraciones es un aspecto que puede ser controlado por el usuario del modelo durante la ejecución del mismo, a través de la variable "Planificación de HU por iteración". Esta variable permite indicar en qué momento del desarrollo de la versión se planifican las iteraciones y qué cantidad de HU se planifican para cada una de ellas.

Otra de las variables importantes del modelo es la "Cantidad de HU por desarrollar", la cual fue inicializada al comenzar el proyecto con 16 HU, luego debido a lo mencionado anteriormente este valor desciende a 12, cuando el cliente elimina 4 HU del ámbito de la versión, al iniciar el desarrollo del proyecto. De aquí en más, este nivel se reduce cada vez que se planifica una iteración (Figura 1), disminuyendo de acuerdo con la cantidad de HU planificadas. Esto puede ser comprendido mejor observando el gráfico "Cantidad de HU por desarrollar" de la Figura 2.

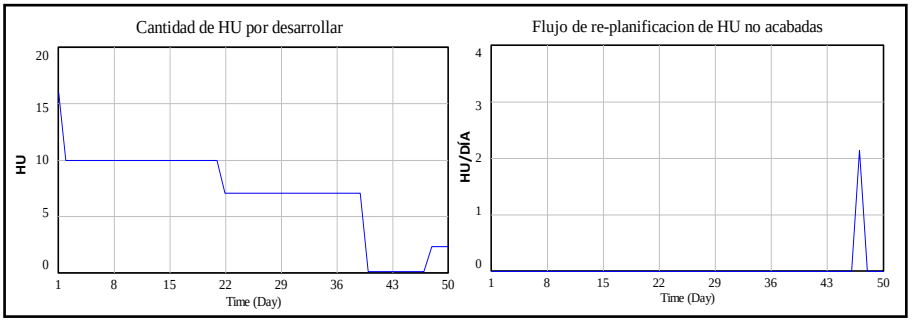

Fig. 2. Cantidad de HU por desarrollar y Re-planificación de HU no acabadas

De esta forma, y como puede verse en la Figura 2, la "Cantidad de HU por desarrollar" desciende a 10 el día 2 del proyecto luego de haberse planificado la primera iteración con 2 HU, el primer día del desarrollo del proyecto.

Esta situación se repite para las otras dos iteraciones de la versión. No obstante, una situación particular que contempla el modelo y que se observa en esta figura, es que al final de la tercera iteración la "Cantidad de HU por desarrollar" aumenta.

Este comportamiento se debe a que, según los datos reales obtenidos del proyecto en estudio, la tercera iteración no llega a completarse por falta de tiempo, por lo tanto, las HU no acabadas deben ser re-planificadas para iteraciones posteriores.

Esta re-planificación ocurre a través del "Flujo de re-planificación de HU no acabadas", que incrementa el nivel de "Cantidad de HU por desarrollar" el día 47, como puede verse en la Figura 2.

Finalmente, luego de pasar por todas las etapas del desarrollo, las HU llegan a la última etapa representada por la variable "HU aceptadas por el cliente". Ésta variable es un nivel que representa la acumulación de HU entregadas al cliente, aceptadas por el mismo y por lo tanto finalizadas. Este nivel puede observarse en la Figura 3.

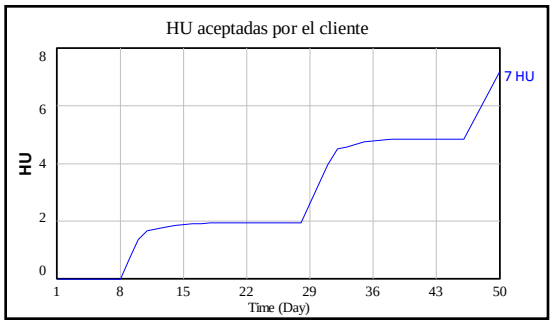

Fig. 3. HU aceptadas por el cliente.

En la Figura 3 se puede ver también que tal como se ha planificado en el proyecto real, la versión es entregada aproximadamente en el día 47, con tres iteraciones realizadas y 7 HU completadas. Las HU no acabadas en la tercera iteración, fueron re-planificadas, como ya se ha explicado anteriormente.

Por otro lado, el segundo caso de validación del modelo corresponde al Sistema "ONess, un Proyecto Open Source para el Negocio Textil Mayorista" cuyos detalles se pueden encontrar en [13].

De la misma manera, la tercera validación se realizó con un "Caso Práctico de Aplicación de la Metodología XP al Desarrollo de Software para un Mini mercado". Los detalles de este caso se pueden encontrar en [14].

4 Experimentos Realizados

El experimento presentado a continuación engloba el desarrollo de una versión de un proyecto XP con los siguientes datos:

- Tamaño de la versión: 18 HU.
- Cantidad de iteraciones: 3.
- Cantidad de HU por iteración:
  - Iteración 1: 8 HU.
  - Iteración 2: 5 HU.
  - Iteración 3: 5 HU.
- Duración de las iteraciones: 15 días.

- Tiempo pactado para la entrega de la versión: 45 días.
- Cantidad de Programadores Novatos: 4.
- Cantidad de Programadores Expertos: 8
- Factor de Carga de Programadores Novatos: 6.
- Factor de Carga de Programadores Expertos: 2.

Luego de ejecutar el modelo con las condiciones de inicio detalladas anteriormente, se obtuvo como resultado que de las 18 HU planificadas para la versión solamente llegaron a entregarse al cliente 17 HU a los 45 días.

Esta situación se debe a que en la primera iteración no se desarrollaron las 8 HU planificadas, lo que ocasionó que 1 HU sea re-planificada para la iteración siguiente, cuya planificación contemplaba 5 HU únicamente al igual que la tercera iteración. Por lo tanto, 1 HU no llega a desarrollarse y entregarse al cliente al final de la versión.

Ante este problema se pueden visualizar dos alternativas que podría evaluar el administrador del proyecto a través del modelo presentado. La primera de ellas (Alternativa 1) contempla un reajuste en la cantidad de horas de trabajo por día de los programadores, en la cantidad de programadores en el equipo de desarrollo o en el FC de los programadores, sin modificar la planificación de la versión.

Por otro lado, la segunda alternativa posible (Alternativa 2) sería un reajuste en la planificación de la versión, es decir, una redistribución de las HU a desarrollar en las iteraciones o una modificación en la fechas de inicio y fin de las mismas. Se debe destacar que esta segunda alternativa, en caso de ser la elegida, implicaría una re-negociación con el cliente, debido a que en XP es él quien decide qué requerimientos deben ser entregados y cuándo.

De acuerdo con lo anterior, en la Figura 4 se pueden observar los 4 Experimentos realizados correspondientes a la Alternativa 1. De esta forma, podemos ver que el Experimento 1 corresponde a la situación inicial mencionada anteriormente, donde al final de la versión se tienen solamente 17 HU aceptadas por el cliente.

Por otro lado, en el Experimento 2 se ha evaluado la incorporación de 1 hora extra a las horas de trabajo por día de los programadores. De acuerdo con ello, se puede observar en la Figura 4 que el ritmo de desarrollo aumenta significativamente debido a que el esfuerzo disponible en horas/programador por día es mayor.

En el Experimento 3 en cambio, se ha modificado la cantidad de programadores novatos de 4 para 3 y la de expertos de 8 para 9, de esta forma de acuerdo con los principios bien conocidos de la Ley de Brooks [15], al modificarse la cantidad de programadores expertos, el factor de carga de los mismos debió aumentarse de 2 a 2.5, lo que supone menor tiempo destinado al desarrollo en sí por parte de estos programadores. De acuerdo con esto, en la Figura 4 puede observarse que el ritmo de desarrollo disminuye con respecto al Experimento 1.

Finalmente, en el Experimento 4 se redujo el FC de los programadores expertos de 2 para 1.5, lo que significa que los mismos destinarán más tiempo de trabajo al desarrollo en sí. Teniendo en cuenta esta modificación en la Figura 4 se puede observar que el ritmo de desarrollo aumenta con respecto al Experimento 1, aproximándose al ritmo alcanzado con el Experimento 2, a medida que se avanza en las iteraciones. Este comportamiento se debe a que, durante el transcurso del tiempo del proyecto, los programadores novatos van adquiriendo experiencia, y por lo tanto, la cantidad de programadores expertos aumenta hacia el final de la versión.

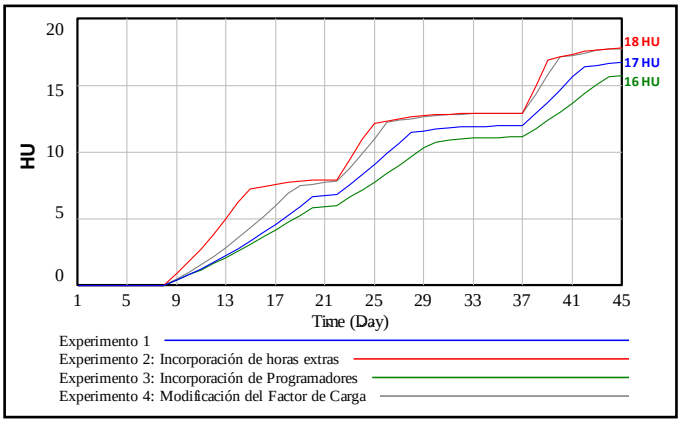

Fig. 4. HU aceptadas por el cliente – Alternativa 1

Luego de haber presentado estos 4 Experimentos se puede concluir que la mejor decisión en este caso sería incorporar horas extras a las horas de trabajo por día, hasta que el problema sea corregido, o bien disminuir el FC de los programadores expertos, lo que arroja mejores resultados hacia el final de la versión.

No obstante lo anterior, se debe decir que los Experimentos 2 y 4 solucionan el problema inicial de re-planificación de 1 HU al final de la primera iteración, tal como se puede observar en la Figura 5. En cambio, el Experimento 3 empeora aún este problema incrementando la cantidad de HU re-planificadas.

Cabe mencionar también que en los Experimentos anteriores los ajustes han sido realizados para la versión completa, sin embargo el administrador de proyectos podría además evaluar cuál es el comportamiento del modelo aplicando estos ajustes solamente a la primera iteración, que es en este caso la que presenta el problema de "HU no acabadas".

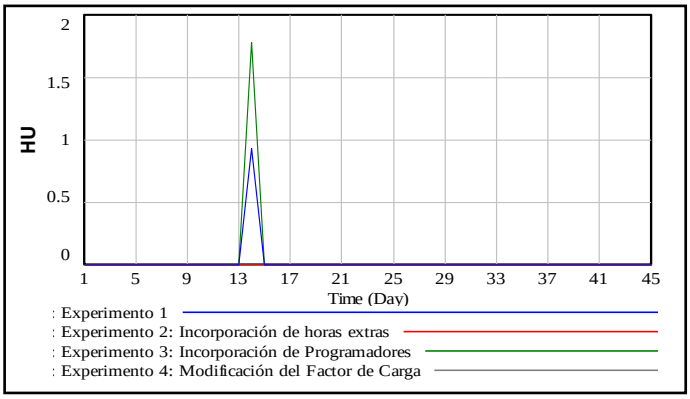

Fig. 5. Flujo de re-planificación de HU no acabadas – Alternativa 1

Por otra parte, de acuerdo con la Alternativa 2 propuesta anteriormente para el caso presentado, el administrador del proyecto podría negociar con el cliente al respecto de contemplar cambios en la planificación de las iteraciones, con el objetivo de evitar la re-planificación de HU no acabadas al final de la primera iteración. De esta forma, se podrían planificar nuevamente las iteraciones pero esta vez de forma más equitativa tal como se detalla a continuación (Experimento 5):

- Iteración 1: 6 HU.
- Iteración 2: 6 HU.
- Iteración 3: 6 HU.

En la Figura 6 se puede observar esta re distribución de HU planificadas en las iteraciones para el Experimento 5 en comparación con el caso base (Experimento 1).

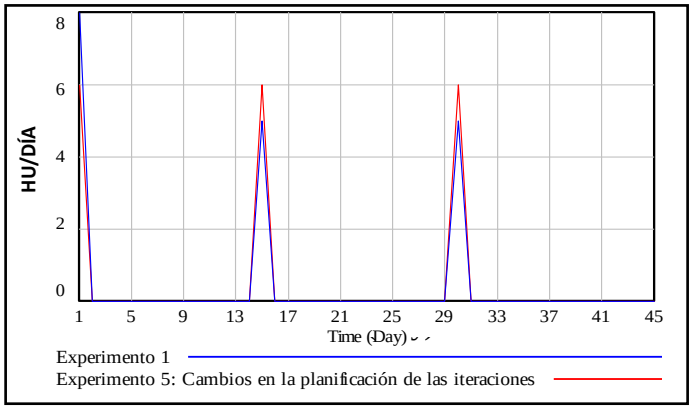

Fig. 6. Flujo de planificación de HU – Alternativa 2

De acuerdo con estos cambios realizados en la planificación de las iteraciones, al disminuir la cantidad de HU en la primera iteración la re-planificación de HU no acabadas desaparece y al final de la versión las 18 HU planificadas son entregadas al cliente. Esto se puede observar en la Figura 7.

En este caso (Experimento 5) se planificó nuevamente la cantidad de HU para cada iteración, no obstante, el administrador podría también evaluar a través del modelo, los resultados de modificar las fechas de entrega de las iteraciones. En este sentido, en el Experimento 6 se incorporaron 5 días más al plazo la entrega de la primera iteración (del caso base) con lo cual, tal como se puede ver en la Figura 7 tampoco se re-planifican HU puesto que todas se alcanzan a desarrollar con 5 días más en la primera iteración.

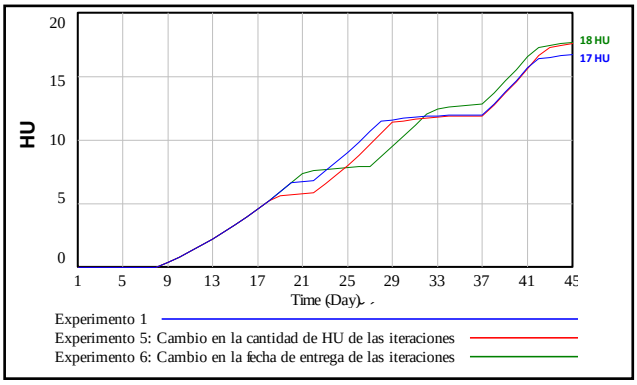

Fig. 7. HU aceptadas por el cliente – Alternativa 2

5 Conclusiones

En el presente trabajo se ha presentado un "Modelo Dinámico de Simulación para Proyectos de Software llevados a cabo con Programación Extrema", que permite a los administradores analizar el efecto del uso de prácticas XP en la gestión de los mismos en diferentes escenarios. Este modelo es una ampliación del presentado en [8] y a diferencia del anterior incluye todos los subsistemas mencionados en la sección 2.

El modelo ofrece buena flexibilidad al usuario del mismo, a través de la modificación de valores de los parámetros durante su ejecución. Entre los valores posibles de modificar se tiene el tiempo de inicio y fin de cada iteración, la cantidad de requerimientos planificados, la cantidad de programadores en el proyecto, el factor de carga y la cantidad de horas de trabajo de los mismos, el porcentaje de tiempo destinado a cada etapa del desarrollo, entre otros.

La validación del modelo, que fue realizada con los casos [12], [13] y [14] pertenecientes a proyectos académicos, fue positiva debido a que el modelo se comportó adecuadamente arrojando resultados que coincidieron con los datos reales de validación. De esta forma, luego de validar el modelo, ejecutarlo bajo diferentes escenarios y realizar el análisis de sensibilidad, se ha llegado a la conclusión de que el mismo cumple con sus objetivos y puede ser utilizado como herramienta para evaluar diferentes decisiones de gestión sobre proyectos de software desarrollados con XP.

6 Trabajos Futuros

Como trabajos futuros se plantea la posibilidad de desarrollar un modelo íntegro que permita simular el desarrollo completo de un Proyecto XP, es decir, incluyendo todas las versiones en que se divide el mismo, y que además contemple la "Planificación por Alcance" que plantea la metodología XP, debido a que el modelo presentado fue construido basándose en la "Planificación por Tiempo" únicamente.

Además de lo anterior, cabe mencionar que existe un gran desafío por alcanzar en el ámbito de los Modelos Dinámicos de Simulación para Proyectos de Desarrollo de Software, y tiene que ver con la construcción de una jerarquía de Modelos Dinámicos

para simular Proyectos de Software llevados a cabo con Metodologías Ágiles. Dicha jerarquía de modelos podría estar inspirada en la presentada en [3] para Proyectos de Software desarrollados con Metodologías Tradicionales, la cual divide a los modelos en Básicos, Intermedios y Avanzados, de acuerdo con la información que se disponga para ejecutar los mismos.

Así también, se pretende avanzar con la construcción de modelos similares para otras metodologías consideradas ágiles como SCRUM [16] y Crystal Orange [17, 18]

Por otra parte se planea utilizar este simulador en cátedras relacionadas con la Ingeniería de Software, con el fin de entrenar a futuros administradores de proyectos.

Finalmente, para colaborar con cualquier investigación futura en Modelos Dinámicos para Proyecto de Desarrollo de Software es preciso construir bases de datos que contengan datos de proyectos de software reales llevados a cabo. Estas bases de datos serían de gran ayuda para validar y probar los modelos construidos.

Referencias

1. Beck, K., Una Explicación de la Programación Extrema. Aceptar el Cambio, Addison Wesley, España (2002). ISBN: 8-47-829055-9.
2. Abdel – Hamid, T. K., Madnick, S. E.: Software Project Dynamics An Integrate Approach, Prentice, New Jersey (1991). ISBN: 0-13-822040-9.
3. Ramos I., Toro M., Ruiz, M.: Modelo Dinámico Reducido, Informe Técnico: LSI-2001-01, Universidad de Sevilla (2001).
4. Lin, C. Y.: Walking on Battlefields: Tools for Strategic Software Management, American Programmer, vol. 6, No. 5, pp. 33-40. (1993).
5. Smith, B. J., Nguyen, N., Vidale, R. F.: Death of a Software Manager: How to Avoid Career Suicide through Dynamic Software Process Modeling, American Programmer, Vol. 6, No. 5, pp. 10-17. (1993).
6. Yong, Y., Zhou, B.: Evaluating Extreme Programming Effect through System Dynamics Modeling, International Conference on Computational Intelligence and Software Engineering, CiSE 2009, pp. 1-4. (2009).
7. Misic, V., Gevaert, H., Rennie, M.: Extreme Dynamics: Modeling the Extreme Programming Software Development Process, Proceedings of ProSim04 Workshop on Software Process Simulation and Modeling, pp. 237-242. (2004).
8. Kasiak, T., Godoy, D.: Simulación de Proyectos de Software desarrollados con XP: Subsistema de Desarrollo de Tareas, XIV Workshop de Investigadores en Ciencias de la Computación 2012. pp. 572 – 576, Argentina (2012). ISBN 978-950-766-082-5.
9. Ventana System Inc., http://www.vensim.com/
10. Aracil, J.: Dinámica de Sistemas, Isdefe, España (1995). ISBN: 8-46 833802-8.
11. Torrealdea J.: Dinámica de Sistemas. Elementos y Estructuras de un Modelo. Construyendo Modelos. Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco. País Vasco.
12. Universidad Politécnica de Valencia: Departamento de Sistemas Informáticos y Computación, http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/
13. ONess, http://oness.sourceforge.net/proyecto/html/index.html
14. Echeverry Tobón L. M., Delgado Carmona L. E.; "Caso Práctico de la Metodología Ágil XP al Desarrollo de Software", Universidad Tecnológica de Pereira: Facultad de Ingeniería. Colombia (2007).
15. Brooks, F.: The Mythical Man-Month: Essays on Software Engineering, Addison Wesley, Massachusetts (1982). ISBN 0-201-00650-2.
16. Schwaber, K., Beddle, M.: Agile Software Development with Scrum. NJ: Prentice Hall. (2002).
17. Cockburn, A.: Agile Software Development. Addison-Wesley. Boston. (2002)
18. Cockburn, A.: Getting Started: Plan the project by milestones. Surviving Object-Oriented Projects: A manager's Guide. Cap. 4. pp. 77-105. Addison-Wesley. (1998)