PROPUESTA DE MODELO DINÁMICO PARA LA MEJORA DE LOS PROYECTOS DE SOFTWARE

Moreno, L.(p); Ramos, I.; Ruiz, M.

Abstract

Software project's success require to deal with the increasing complexity and dynamic characteristic of software development and software process changing. In order to improvement software projects, a dynamic simulation model is proposed to captures the dynamic of software projects in order to improvement the software process and the capability of the organizations according with the major standards of quality and software assessment. Therefore, new key feedback dynamics from previous models of project dynamics are identified as dynamic hypothesis. Also, metrics accordingly with software process capability has been proposed. A formal model in VenSim language is presented. The calibration of the model into a real project under construction, and the partials test might indicate the adequacy to simulate the dynamics of the project.

Keywords: Modeling, software process improvement, project simulation, software engineering

Resumen

Los proyectos de Software se caracterizan por su enorme complejidad y por sus dinámicas complejas. En orden a mejorar los proyectos de software se va a presentar una propuesta de un modelo para contemplar las dinámicas de los Proyectos Software con objeto de mejorar los procesos y capacidad de las organizaciones de acuerdo con los principales estándares de calidad y evaluaciones. Asimismo, a partir de modelos anteriores se van a identificar nuevas dinámicas claves; además, se va a proponer las métricas necesarias según las capacidad de los procesos. Su calibración y los tests necesarios van a indicar la adecuación para simular la dinámica de los proyectos.

Palabras clave: Modelado, mejora de procesos de software, simulación de proyectos, ingeniería de software.

1. Introducción

Los proyectos de desarrollo complejos son difíciles de gestionar, particularmente por la naturaleza dinámica de los sistemas de proyectos [1]. Procesos iterativos, recursos y gestión, interactúan para generar comportamientos que pueden ser imprevistos. Así, por ejemplo, el progreso puede disminuir, incluso pararse, o volverse negativo sin ninguna razón aparente [2]. Modos de comportamiento tales como el aumento y decrecimiento del porcentaje del trabajo completado, e incremento en el *backlog* del proyecto, pueden ser característicos de proyectos que añaden nuevas tareas en unas ciertas estructuras, conocidas como "*tipping points*"; en donde los gestores del proyecto pueden perder el control de la acumulación de estas nuevas tareas y sus proyectos, si éstos se mueven por debajo de condiciones de tipping point.

Los comportamientos de sistemas multi-proyecto y su gestión se han descrito también por medio de estas estructuras dinámicas. Así, Repenning y Black [3] investigaron "*fire-fighting*" (asignación no planificada de ingenieros y otros recursos para resolver problemas tardíos)

en una serie de proyectos, parcialmente simultáneos, de desarrollo de productos. Las condiciones de Tipping point configuraban un equilibrio inestable, donde pequeñas perturbaciones, empujaban al sistema hacia uno o dos equilibrios estables. Estructuras y condiciones similares han sido muy estudiadas en orden a la determinación de modelos dinámicos genéricos que ayuden a comprender las causas de fallos en los proyectos.

Los Proyectos de Software se caracterizan por su enorme complejidad e incertidumbre, así como por su constante evolución en orden a cubrir los diversos tipos, tamaños y complejidades, del software desarrollado. Sin embargo, estos factores dinámicos no han sido explícitamente señalados en las herramientas tradicionales de control y planificación, y los problemas encontrados en el desarrollo de software son tratados de una manera estática.

La importancia de las dinámicas de los procesos es difícil de rebatir debido a los bien conocidos (y frecuentemente ignorados) efectos combinados de presión de plazo, sobrecarga de comunicación, condiciones cambiantes de los negocios, volatilidad en los requisitos, condiciones y peticiones de usuario, experiencia, métodos de trabajo tales como revisiones y actividades aseguradoras de la calidad, subestimación de tareas, retrasos burocráticos, cambios organizacionales, acontecimientos desmotivadores y otros fenómenos socio-técnicos, y los mecanismos de realimentación entre ellos (feedbacks) [5]. Estos efectos, complejos e interactivos, son acertadamente modelados a través de los modelos dinámicos. El conocimiento de la interrelación entre los distintos factores técnicos, sociales e interorganizativos, acoplados a través de herramientas de simulación, proveerá un medio *para mejorar sus procesos.*

Los modelos de simulación se han usado durante mucho tiempo en diversas industrias tales como automoción, química, manufacturación y tecnologías de la información, para predecir la ejecución y rendimiento de los sistemas complejos.

La Dinámica de Sistemas se ha aplicado a los procesos de software, comenzando con el modelo de referencia debido a Abdel-Hamid y Madnick [4] en 1991, en el estudio de las dinámicas de los proyectos de software para un simple ciclo en cascada (waterfall cycle). Desde entonces, ha habido innumerables publicaciones con el empleo de dinámica de sistemas [5][6] y se han considerado aspectos de gestión estratégica de desarrollo de software [1], soporte de gestión de apoyo a los procesos de mejora [7], programas de entrenamiento de gestión de proyectos, y desarrollo e integración de COTS (commercial-off-the-shelf) [8]. Actualmente se focaliza en cuestiones de los procesos SISOs (Software-Intensive Systems of Systems), tales como acometer los rápidos cambios asegurando simultáneamente la alta dependencia con modelos dinámicos para valorar los procesos híbridos incrementales [9]; soporte a la toma de decisión y planificación de versiones en desarrollo de software incremental con análisis de estabilidad [11]; evaluación de tareas globales de software a través de la simulación [21] y el uso de la simulación para análisis de casos de valores añadidos en los procesos de Negocios [26]. En definitiva, un área de investigación que dirige su atención a la predicción de impacto de las numerosas cuestiones que se proponen en los procesos de software y en la simulación de modelos.

En los siguientes apartados se expondrá los objetivos y metodología de nuestro trabajo, se realizará una revisión y comparativa de los principales modelos dinámicos aplicados a la Ingeniería de Software, seguirá una sección dedicada a la justificación del modelo propuesto, después se explicará los detalles del modelo, y finalmente, se expondrá las conclusiones y trabajos futuros.

2. Objetivos

El objetivo principal del trabajo es contribuir a la mejora de la gestión de los Proyectos de Software por medio del modelado dinámico, utilizando técnicas de dinámica de sistemas y modelos previos, con objeto de proveer un marco de comprensión de los problemas y cuestiones en los proyectos de software; así como, valorar y optimizar las estrategias de los procesos desde un punto de vista sistemático.

Comprender las dinámicas del desarrollo de software implica:

- *Mejorar y ayudar en la toma de decisiones*. El conocimiento de las consecuencias de las decisiones (ya sean sobre los proyectos o sobre las políticas organizativas) mejorará al disponer del conocimiento causal y dinámico de los procesos que están involucrados. Entre estas decisiones se incluyen el establecimiento de los presupuestos de los proyectos y plazos, análisis del retorno de inversión, compromisos coste/plazo/calidad u otros factores, contratación de personal, decisiones de gestión en riesgos, estrategias de mejora de proyectos, etc.
- *Reflejar o capturar la dinámica de los proyectos*. Los distintos modelos dinámicos se han construido integrando y/o adaptando distintos factores dinámicos de modelos previos con nuevas dinámicas específicas de los proyectos en cuestión.
- *Estimular la comprensión y aprendizaje*. El proceso de modelado crea un marco conceptual para la comprensión del comportamiento del proyecto.
- *Ayudar como un mecanismo de comunicación* en las distintas unidades de las organizaciones.
- *Promover la mejora de Procesos dentro de las Organizaciones.* El análisis causal, el modelado y la simulación suponen cuantificar los procesos y mejora continua, y se sitúa en los niveles más altos de los estándares de madurez de organizaciones, por ejemplo CMMi [27].
- *Calidad de Software*. Para definir la calidad del software y establecer las metas de calidad, es importante tener un conjunto de métricas bien definidas en los productos software. Un paso necesario para identificar un conjunto relevante de métricas es comprender los mecanismos en el proyecto de software. Tal comprensión puede ser lograda por medio de la construcción de modelos de simulación de los procesos de desarrollo de software.
- *Mostrar los bloques de construcción básicos y modelar las infraestructuras para los procesos de desarrollo de software***.**

3. Metodología

En este trabajo nos basamos en la metodología de Sistemas Dinámicos, Pensamiento de Sistemas (Thinking Systems), así como los estándares de Mejoras y Procesos Software, y modelos dinámicos previos para exponer cuestiones planteadas en los objetivos. Para ello, se ha realizado una revisión de la literatura para poder situar el estado actual y cuestiones de las dinámicas de software y se han seguido los pasos propuestos por Kellner, Madachy y Raffo [25] para la construcción del modelo dinámico.

Se ha utilizado Vensim® 5.0 como herramienta de modelado y también para la construcción de los diagramas causales y de los diagramas de flujos y niveles.

4. Dinámica de Sistemas y los Proyectos de Software

La Dinámica de Sistemas es una metodología empleada para comprender cómo los sistemas cambian a lo largo de tiempo, siendo un sistema una colección de elementos que interactúan continuamente en el tiempo, para formar una unidad completa [19]. Fue desarrollada para aplicar la teoría de control a los análisis de los sistemas industriales, económicos y sociales. Provee un marco, integrado y rico, para capturar los fenómenos, los procesos, y sus interrelaciones; así como es muy adecuada para tratar con la complejidad de los sistemas de software, porque captura las dinámicas, los bucles de realimentación y los fenómenos que interactúan y causan los comportamientos complejo que se observan en el mundo real [5][12]. La figura 1 muestra la representación de un modelo de proyecto [1] como una compleja integración entre las dinámicas de planificación y control, recursos humanos, y producción de software; reflejándose los elementos y subsistemas que lo conforman junto con los bucles de realimentación.

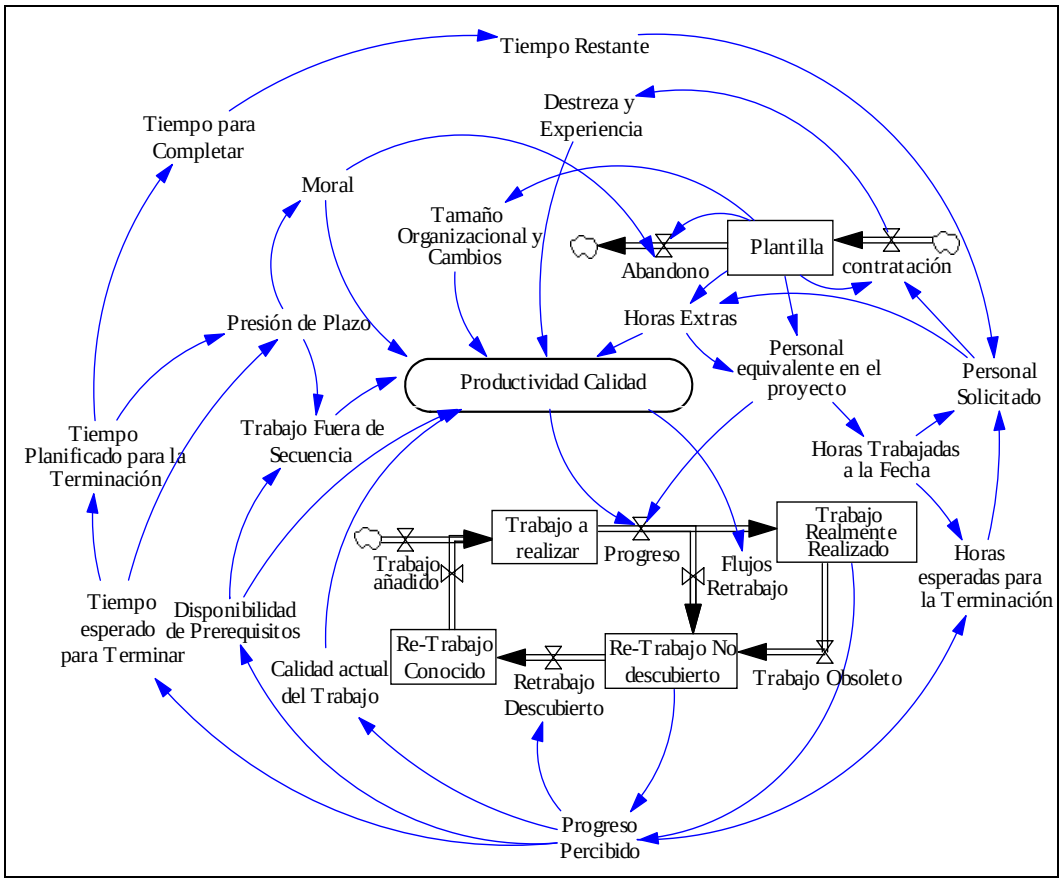

Figura 1. Dinámicas de Proyecto [1].

Entre las estructuras subyacentes que afloran en la figura 1, están:

- La estructura del trabajo desarrollado. Contiene los componentes asociados con la producción de software y los niveles de trabajo debidos a errores, por no haber pasado los criterios de calidad; por lo que tendrán que volverse a revisar. Esta estructura ha sido muy estudiada y casi todos los proyectos de software tienen su ciclo de trabajo. El efecto del re-trabajo es el responsable del desplazamiento y sobrecarga del alcance del proyecto, y extensión en la duración del proyecto.

- Efectos de realimentación en la productividad y calidad del trabajo. La productividad y la calidad son afectadas por la *calidad actual del trabajo*, *disponibilidad de prerrequisitos, trabajo fuera de secuencia, presión de plazo, motivación, destreza y experiencia, cambios organizacionales y horas extras.* Así, el trabajo fuera de secuencia causa una reducción en la productividad y calidad, como resultado de los errores cometidos en los productos incompletos o realizados incorrectamente en las etapas precedentes. Estos efectos, junto a la combinación de planes inadecuados, como los derivados típicamente del arranque del proyecto, y/o cambios a mitad del proyecto, pueden causar problemas en los momentos iniciales o medios de las etapas del proyecto, y se incrementan al final.
- Planificación y control. Establece las estimaciones iniciales y revisiones necesarias del proyecto hasta que termina. Contiene elementos para estimar la plantilla deseada y la fecha de finalización. El sector de control compara las medidas actuales con las estimadas y comunica las evaluaciones a otros segmentos del modelo para reajustar sus parámetros.
- Estructura de plantilla de personal. Contiene los elementos necesarios para la contratación, formación y transferencia de personal en el proyecto.

La construcción de los modelos de proyecto y procesos de software se facilita por medio de *procesos genéricos de flujos*, que son bloques constructivos a través de las cuales se van formando estructuras de orden superior: las *infraestructuras,* estructuras más grandes que están compuestas por varias microestructuras, que producen comportamientos más complejos; y las *cadenas de flujo,* que son secuencias de flujos y niveles que frecuentemente forman el esqueleto de una porción del modelo [13].

5. Bases y Motivación del Modelo Propuesto

Los artefactos de software, que fluyen a través de los modelos de software estudiados, son definidos generalmente como “tareas” que se asumen intercambiables y uniformes en tamaño; considerándose en una cadena de producción de software en los estados de tareas desarrolladas, revisadas en el sistema de calidad y, finalmente, pasadas el proceso de test.

Esta abstracción en el flujo de tareas, es más apropiada dentro de una perspectiva de gestión de alto nivel, siendo difícil su empleo como herramienta de planificación o de seguimiento de hitos del proyecto.

También, muchas de las funciones de control del proyecto usan un indicador de progreso que puede dar lugar a confusión. Las estructuras de planificación y control no describen todos los entornos, especialmente los más modernos. La definición de las actividades de calidad no es estándar.

Ford y Sterman usaron el método de elicitación de conocimiento descrito en [14], para estimar importantes funciones no lineales en un modelo de desarrollo de producto [15]. El modelo desagregó el proyecto global en un número de fases. Una estructura genérica servía como referencia para representar cada fase, que se concretaba con sus parámetros específicos. Las fases eran mutuamente dependientes entre sí en la información necesaria para completar el trabajo, estableciéndose restricciones entre ellas, a través de *relaciones de precedencias,* que caracterizan el grado de condiciones por el cual las tareas pueden realizarse en paralelo o deben hacerlo secuencialmente. Las *relaciones de precedencia externa* describen las dependencias de las tareas de desarrollo de una fase a la otra, mientras que las *relaciones de precedencia interna* describen las interdependencias de las tareas de desarrollo dentro de una sola fase.

Madachy en [16] utilizó la concurrencia para proveer un marco robusto para modelar los procesos de software y sus restricciones, y servir para evaluar las estrategias en la

introducción de nuevos procesos, metodologías y herramientas. Modeló relaciones de concurrencia externa de procesos, relaciones que exhiben un alto grado de paralelismo e influencia entre fases, y relaciones que no están disponibles en métodos PERT/CPM.

Moonseo y Peña-Mora [17] presentaron un modelo para analizar los efectos de los errores y cambios que afectan al rendimiento del proyecto. El modelo incluía la gestión de calidad, procesos de petición de información (Request for Information) y los procesos de toma de decisiones. El modelo del proyecto consistía de una *estructura de proceso* y cuatro *estructuras de soporte*: alcance del proyecto, adquisición y asignación de recursos, rendimiento del proyecto y políticas. Para tener la flexibilidad de las herramientas de planificación tradicionales, tales como CPM, PDM y PERT; el modelo incorporaba la lógica de estas herramientas. Por ejemplo, el modelo trata las típicas relaciones de precedencia de estas herramientas (comienzo-comienzo, comienzo-final, final-comienzo, final-final) como concurrencias externas. Mientras las dependencias del trabajo dentro de una actividad, se representa por medio de una variable exógena de concurrencia interna.

Lee y Peña-Mora [18] analizaron los efectos del re-trabajo y cambios en la construcción, y propusieron un modelo dinámico que integra los métodos *de facto* en la construcción con los modelos de dinámica de sistemas. Estos modelos [17][18] fueron desarrollados para acometer los cambios continuos y toma de decisión en los proyectos de construcción, tanto a nivel estratégico como operativos.

En [6], se presentó un modelo de patrones genéricos de mejora, propuestos previamente por Salge [20] en manufactura. Estos patrones de mejoras actuarían sobre las distintas variables del modelo correspondiente y se reflejarían en los valores iniciales concretos, tiempos de vida media, tiempos de erosión y esfuerzos para la mejora.

Con base en las referencias anteriores, creemos que un modelo que pueda ayudarnos a resolver las cuestiones de los proyectos de software a nivel estratégico, operativo, de planificación, control y mejora, entre otras, y con un ámbito amplio de aplicación, debe basarse en el estudio de los procesos o actividades al nivel más desagregado posible. Establecer los límites del modelo, lo que está incluido o excluido del modelo, es una tarea necesaria e iterativa, así como la elección del lenguaje de simulación. Las cuestiones claves que se vayan a plantear van a tener una influencia importante en la elección de los parámetros del modelo, y las variables de entrada y salida [25].

6. Modelo Propuesto

Propuestos los fundamentos en la sección anterior y tomando como referencia los modelos tratados, nuestro objetivo es diseñar una *estructura genérica de actividades*, que junto a las estructuras de mejoras y soporte permitan tratar las peculiaridades de los proyectos de software.

La figura 2 muestra el esquema general del modelo, adaptado de Moonseo y Peña-Mora [17], que consiste en una estructura general de actividades y las estructuras de soporte y mejora; siendo la Actividad Genérica la parte que va a tener una especial consideración.

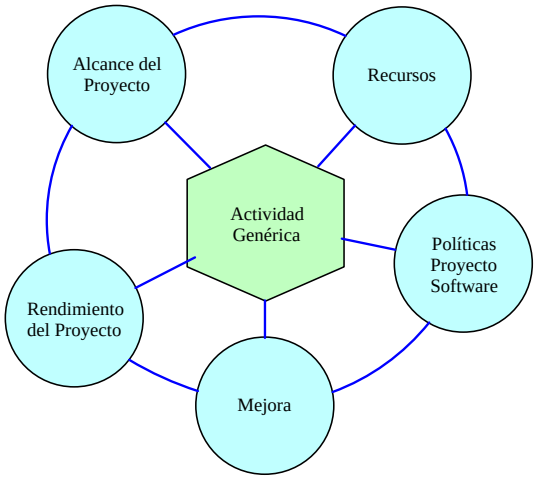

Figura 2. Esquema Modelo de dinámica de actividades adaptado de [17].

La figura 3 ilustra, simplificadamente, la *estructura genérica de trabajo*, adaptada de Lee y Peña-Mora [18]. En el modelo, el flujo de Trabajo se representa como tareas que fluyen hacia o desde siete niveles principales: Tareas a Realizar, Tareas en Espera de Coordinación, Tareas Pendientes de Solución de Coordinación, Tareas Esperando Gestión de Calidad, Tareas Completadas, Tareas a Revisar y Tareas en espera de Decisiones de Cambio.

Las tareas disponibles en un tiempo dado son introducidas en el nivel *Tareas a Realizar* a través del flujo *Flujo de introducción de Tareas*. Las tareas se van completando a través del *Flujo de Tareas,* acumulándose posteriormente en el nivel *Tareas Esperando Gestión de Calidad.* Dependiendo de la calidad, algunas de ellas tendrán que pasar un proceso de revisión en *Tareas a Revisar;* en donde se sabrá las tareas que se han de volver a tratar, y acumularse de nuevo en el nivel *Tareas a Realizar.* Aquellas tareas que hayan superado los criterios de calidad, pasarán al nivel *Tareas Completadas.* Estas tareas, bajo ciertas condiciones, pueden volver al nivel *Tareas a Realizar.*

Si durante el desarrollo de las tareas fuera necesario la comunicación o coordinación con actividades precedentes, debido por ejemplo a errores encontrados, las tareas fluirían desde y hasta *Tareas a Realizar* a través de: *Flujo de Peticiones de Información, Flujo Petición Predecesor* y *Flujo de Resolución de tareas pendientes.* Los otros dos niveles involucrados (*Tareas en Espera de Coordinación* y *Tareas Pendientes Solución Coordinación*) representan distintos estados de las tareas pendientes, en espera de la solución del proceso de coordinación.

En la figura 3 aparece también la variable *Flujos de Cambios,* cuando en una actividad es necesario decidir sobre un proceso de cambios y su resolución. Asimismo, las tareas en el nivel *Tareas en espera de Coordinación*, en ciertos casos, es posible que pasen, por medio del flujo *Coordinación a Cambios,* al nivel *Tareas en Espera de Decisiones de Cambios*.

Se van a considerar, por consiguiente las siguientes propuestas de mejora y adaptaciones al modelo de [18], para proyectos de software:

1. *Tareas a Revisar.* Hemos añadido este nivel (color verde en la figura 3). Es necesario revisar muchas de las actividades de software en que se requiere programación, después de pasar las etapas de calidad. Este nivel estaba ausente de las estructuras de los modelos de referencia, ya que sus ciclos correspondientes tienen una duración breve, y la construcción es una manifestación física.

2. La concurrencia externa es unos de los elementos a adaptar a las dinámicas de software. Aunque se ha sugerido formas de la curva de relaciones entre arquitectura de software y requerimientos [22], y en proyectos RAD (Rapid Application Development) [13], hay que definir nuevas concurrencias externas e internas para tratar las derivadas de los actividades de procesos globales y procesos distribuidos [21].
3. Estabilidad de tareas. Debido a cambios en el alcance del proyecto de software o cambio de requisitos.
4. Coordinación y Comunicación entre equipos. Típico para Proyectos de Software actuales [24]. En los proyectos de software cuando los equipos están separados en tiempo y distancia, la efectividad de la comunicación se deteriora tanto en calidad y oportunidad.
5. Mejora a través de la vida media de los parámetros del modelo. En esta estructura se definirá la mejora a través de los tiempos de mejora de vida media y deterioro en las variables exógenas correspondientes del modelo.
6. Funciones de control y extensibilidad a Fases y/o Módulos. A través de las funciones de arrays de Vensim, se puede extender el control por medio de los valores de actividades. Así, tenemos funciones globales del proyecto, fases y módulos.

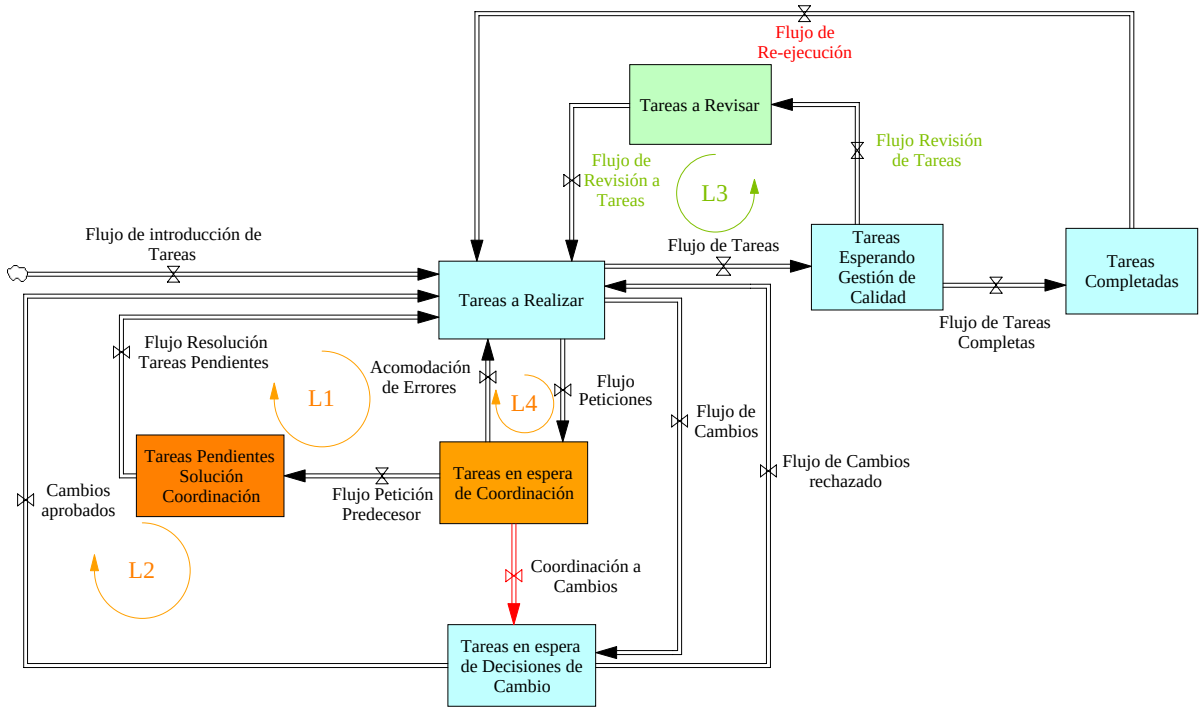

Figura 3. Modelo de dinámica de actividades adaptado de [18].

Además de los bucles de realimentación ya comentados, la figura 3 muestra de color verde, las aportaciones que presentamos en este trabajo y en naranja, aquellas que han sido adaptadas a nuestra propuesta. De este modo, tenemos identificadas importantes *iteraciones sin ningún valor añadido* que ocurren en las actividades de software, por lo que cualquier decisión dentro de éstas ha de ser cuidadosamente estudiada, debido a la estabilidad de las tareas, sensibilidad a cambios y efectos “multiplicadores de trabajo” por cambios (internos o externos).

En la figura 4 aparece la entrada de Tareas a Revisar que hemos incorporado a nuestra propuesta. Las figuras 5 y 6 muestran cómo afectan a las Tareas a Realizar y a las Tareas

Completadas, la incorporación de las Tareas a Revisar en el modelo, adaptado de [18]. Se consideran cuatro actividades con igual número de tareas iniciales (en unidades de trabajo – *UT*). Conforme el tiempo avanza van disminuyendo las Tareas a Realizar (a1, a2, a3, a4) y aumentado el número de las Tareas Completadas (figura 5), observándose la simultaneidad de las tareas en las etapas finales de cada una y las Tareas a Revisar en las etapas medias-altas del desarrollo de las actividades.

La influencia en el desarrollo del proyecto de la introducción de este nivel vendrá, en gran medida, determinada por el valor del *Flujo de Tareas,* que viene determinado por el Tiempo disponible para cada actividad y el valor de la disponibilidad de Fuerza de Trabajo. En el caso de estudio realizado, se muestra un ligero incremento en el tiempo de ejecución de las actividades asociadas (respecto al modelo sin Tareas en Revisión), no porque haya aumentado significativamente el número de Tareas a Realizar, sino por la incorporación del retraso asociado a la adición de un nivel al modelo.

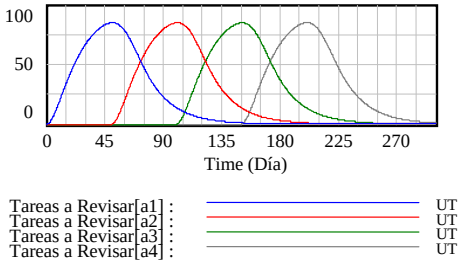

Figura 4. Evolución de las Tareas a Revisar.

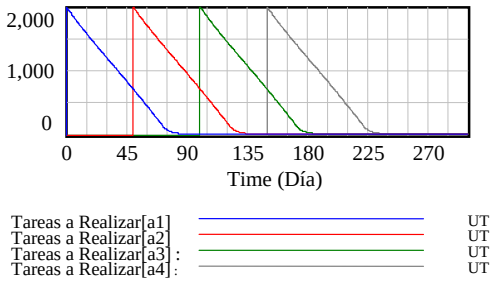

Figura 5. Evolución de las Tareas a Realizar.

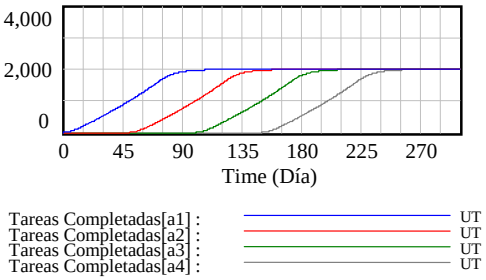

Figura 6. Evolución de las Tareas Completadas.

Aunque la introducción del nivel *Tareas a Revisar* puede suponer un retraso debido a la carga de trabajo adicional, supone un *stock o buffer* proactivo para estimar la duración de los proyectos, que hace más realista dicha estimación.

7. Resultados y Conclusiones

Nuestro trabajo viene de la necesidad de mejorar la comprensión de los proyectos de software a través del modelado dinámico. Se ha realizado una revisión de la literatura, así como un estudio de las estructuras y subsistemas que integran los procesos y proyectos de software, junto con las iteraciones sin ningún valor añadido que tienen lugar, y sugerimos el enfoque de los modelos hacia las "actividades" que se desarrollan en el mismo.

Hemos propuesto mejoras para proyectos de software partiendo de los modelos de referencia estudiados y hemos analizado una propuesta concreta relacionada con la incorporación de las *Tareas a Revisar*. Con las mejoras propuestas, se reduce el nivel de abstracción y se mejora el tratamiento de los procesos y tareas que se desarrollan tanto a nivel de software, como organizativos y estratégicos. Esta representación elegida supondrá una eficaz integración de los modelos dinámicos en el contexto las diversas iniciativas de calidad, como CMMi [27], ISO 9000 [28], ISO 15504 [29], y EFQM [30]; ayudará a concentrarse en las representaciones de sus procesos, en la manera en que están implementados (cómo son, cómo son llevados a la práctica, y cómo son documentados), en la consideración de las distintas futuras implementaciones (como deberían ser) basadas en procesos de actividades de mejoras por aplicación de nuevas herramientas y tecnologías de selección de procesos alternativos. Además, ayudará a la gestión, selección y definición de las medidas de los procesos, a analizar y rediseñar programas de medidas de rendimiento de los procesos corporativos, y al análisis de nuevos ciclos de vida para Proyectos Software.

Referencias

[1] Lyneis F, Cooper K, Els S., "Strategic management of complex projects: a case study using system dynamics". *System Dynamics Review* Vol. 17(3), 2001, pp. 237– 260.

[2] Taylor, T. and Ford, D.N., "Tipping point failure and robustness in single development projects", *System Dynamics Review*, Vol. 22(1), 2006, pp. 51-71.

[3] Black, L. Repenning, N., "Why firefighting is never enough: preserving high-quality product development". *System Dynamics Review* 17(1), 2001, pp.33– 62.

[4] Abdel-Hamid, T. and Madnick, S., "Software Project Dynamics: An Integrated Approach", Prentice Hall, Englewood Cliffs, NJ. 1991.

[5] Madachy, R.J., "Software Process Dynamics", Wiley-IEEE Press**,** 2007.

[6] Moreno, L., Ramos, I., Ruiz, M., "Estado del Arte del Modelado Dinámico para la Mejora de los Proyectos de Software". XI Congreso Internacional de Ingeniería de Proyectos. Congreso Internacional de Ingeniería de Proyectos (11). Num. 11. Lugo, España. Aeipro - Asociación Española de Ingeniería de Proyectos. 2007. Pag. 1-10.

[7] Ruiz, M., "Modelado y Simulación para la Mejora de los Procesos Software", *Tesis Doctoral*, Universidad de Sevilla, 2003.

[8] Kim, W.K. and Baik, J., "Dynamic models for COTS glue code development and COTS integration, University of Southern California, CS599, 1999.

[9] Madachy, R.J., Boehm B. and Lane J.A., **"**Assessing Hybrid Incremental Processes for SISOS Development**,** Software Process Improvement and Practice", 2007, Vol.12, pp. 461–473.

[11] Pfahl D., Al-Emran A. and Ruhe G., "A System Dynamics Simulation Model for Analyzing the Stability of Software Release Plans", Software Process Improvement and Practice". 2007; Vol.12, pp. 475–490.

[12] Aracil, J., "Introducción a la dinámica de sistemas". Alianza Editorial Textos, 1986.

[13] Madachy, R. J. , “Software Process Concurrence”, *Conference Proceeding, The 20th International Conference of the System Dynamics Society,* July 28- August 1, 2002 Palermo, Italy.

[14] Ford D. and Sterman J., “Expert knowledge elicitation to improve formal and mental models”, System Dynamics Review, Vol. 14(4), 1998, pp. 309-340.

[15] Ford D, Sterman J, “Dynamic modeling of product development process”, System Dynamics Review, Vol. 14(1), 1998, pp. 31-68.

[16] Madachy, R., “Reusable Model Structure and Behaviors for Software Processes”. SPW/*ProSim 2006 Workshop May 20, 2006.*

[17] Moonseo, S. and Peña-Mora, F., “Dynamic change management for construction: introducing the change cycle into model-based project management”, *System Dynamics Review*, Vol. 19, 2003, pp. 213–242.

[18] Lee, S. and Peña-Mora, F., "Understanding and managing iterative error and change cycles in construction", *System Dynamics Review*, Vol. 23, 2007, pp. 35–60.

[19] Martin, L. The First Step, MIT**:** 57.1997.

[20] Salge, M., “Exploring Patterns of Process Improvement with a Generic Model”, *Conference Proceeding, The 24th International Conference of the System Dynamics Society,* July 23-27, 2006 Nijmegen, The Netherlands, 2006.

[21] Pfahl, D., Al-Emran A. and Ruhe G., “A System Dynamics Simulation Model for Analyzing the Stability of Software Release Plans”, Software Process Improvement and Practice”*.* 2007; Vol. 12, pp. 475–490.

[22] Fakharzadeh, C., Mehta., N., “Architecture Development Process Dynamics in MBASE”, *Conference Proceeding, The 184th International Conference of the System Dynamics Society, August 6 -10*, 2000, Bergen,Norway, 2000.

[23] Swebok. Guide to the Software Engineering Body. IEEE, Versión 2004.

[24] Espinosa, J.A., Carmel E., “The impact of time separation on coordination in global software teams: a conceptual foundation”. *Software Process Improvement and*

Practice, 2003, 8(4): pp. 249–266.

[25] Kellner, M.I., Madachy, R.J., Raffo, D.M., “Software process simulation modeling: Why?

What? How?”, The Journal of Systems and Software, Vol. 46 (2/3), 1999. pp. 91-105.

[26] Boehm, B., Huang, L.G., “Value-Based Software Engineering: A Case Study”, Computer, v.36 n.3, Marzo 2003, pp.33-41.

[27] Chrissis M.B., Konrad. M, Shrum S., 2003. “CMMI: Guidelines for Process Integration and Product Improvement”, Addison-Wesley: Boston, MA, USA, 2003.

[28] Mutafelija B., Stromberg H., “Systematic Process Improvement using ISO 9001:2000 and CMMI”, Artech House Computing Library. 2003.

[29] ISO/IEC 15504-5. 2006. Information Technology – Process Assessment – Part 5: An Exemplar Process Assessment Model.

[30] European Foundation for Quality Management. http://www.efqm.org

Correspondencia (Para más información contacte con):

Luis Moreno Iglesias.
Servicio de Informática y Comunicaciones.
Universidad de Sevilla.
Avenida de Reina Mercedes s/n, 41012 Sevilla, Spain.
Phone: + 34 95 455 99 46
Fax: + 34 95 455 65 45
E-mail : luism@us.es
URL: http://www.us.es/SIC