Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

Tipo de artículo: Artículos originales
Temática: Ingeniería de software
Recibido: 02/12/2019 | Aceptado: 07/03/2020 | Publicado: 30/03/2020

Simulación del proceso de desarrollo de software: una aproximación con Dinámica de Sistemas y el Método de Larman

Simulation of the software development process: an approximation using System Dynamics and the Larman Method

German Lenin Dugarte Peña [1][0000-0001-9760-7084]*, **Maria Isabel Sánchez Segura** [2][0000-0002-2339-7851], **Fuensanta Medina Dominguez** [3][0000-0002-3249-2834], **Antonio de Amescua** [4][0000-0003-4355-6896]

[1] Departamento de Informática, Universidad Carlos III de Madrid. gdugarte@inf.uc3m.es
[2] Departamento de Informática, Universidad Carlos III de Madrid. misanche@inf.uc3m.es
[3] Departamento de Informática, Universidad Carlos III de Madrid. fmedina@inf.uc3m.es
[4] Departamento de Informática, Universidad Carlos III de Madrid. amescua@inf.uc3m.es

* Autor para correspondencia: gdugarte@inf.uc3m.es

Resumen
Poner en marcha cualquier proyecto de software involucra el consumo de recursos críticos. El ingeniero de software no puede experimentar con procesos de desarrollo sin ponerlos en marcha en proyectos reales, debido al tiempo que ello conlleva y a los elementos implicados, de modo que es importante contar con herramientas para previsualizar el resultado de la ejecución del proceso y cómo las variables de entorno le afectan, buscando anticipar en qué condiciones se va a desplegar el proceso. Este artículo presenta el modelado y simulación de un proceso de desarrollo de software por medio del enfoque de la Dinámica de Sistemas (DS), que permite representar gráficamente los elementos intervinientes en el proceso e incorporar la cantidad relevante de parámetros involucrados. Se tomó como referencia el modelo de estimación de costes COCOMO, que cuenta con una fundamentación teórico-práctica que avala su fiabilidad. Para la construcción del modelo, la referencia de sistema real fue el proceso software de Craig Larman (Método de Larman). El modelo de simulación presentado permite hacer estimaciones iniciales del comportamiento del proceso software, y de los elementos que lo conforman, durante el transcurso de un tiempo de simulación configurable. Se analizan variables de estado del sistema, que permiten concluir sobre efectos de los parámetros en el comportamiento del sistema en general. El modelo deriva en una herramienta de soporte a los equipos de gestión, y a las empresas que hacen de la Gestión de Proyectos Tecnológicos su negocio principal.

Palabras clave: Dinámica de Sistemas, Software Engineering Economics, Gestión del Proceso Software, Método de Craig Larman, Modelado y Simulación de Procesos Software.

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

Abstract
The implementation of any software development process involves the consumption of critical resources. Software engineers cannot experiment with different development processes before starting them in real projects, due to the time that would entail and the number of elements that are involved, so it is vital to have tools that allow the pre-visualization of the results of executing the software development process and how the environmental variables affect it, thus being able to anticipate under what conditions the software development process will be deployed. This paper presents the modelling and simulation of a software development process using System Dynamics (SD), which allows the graphical representation of the elements intervening in the software process, and the incorporation of as many relevant elements as possible. As a software costs estimation reference, the COCOMO estimation model was used; which beyond being reliable has a theoretical-practical foundation. As an ideal, and real, software process system, the Craig Larman Software Process model was chosen, also known as the Larman Method. The simulation model developed here, allows one to make some initial estimation of the software process and its elements' behavior in the course of the simulation time. This is possible thanks to the observation and study of the system's state variables, empowering one to discern about the effect of changes in the parameters on the general process. This model becomes a tool for supporting Software Project Management teams and enterprises whose business care on Technological Projects Management.

Keywords: *Systems Dynamics, Software Engineering Economics, Software Process Management, Craig Larman`s Method, Software Process Simulation Modelling.*

Introducción

En la gestión de procesos de ingeniería de software, cada vez se hace más evidente la necesidad de contar con herramientas que permitan una mayor supervisión y monitorización del proceso en su totalidad, con la idea de prever con antelación cual puede ser el mejor proceso a aplicar a un proyecto concreto según sus características, o qué decisiones se deben tomar en un momento determinado. El tiempo de desarrollo de los proyectos y el consumo de recursos necesarios para el proceso de ingeniería del software se han convertido en los factores clave que son sujeto de observación a la luz de distintas perspectivas o estilos que rigen los procesos de ingeniería de software; es así que se ha comenzado a hablar de técnicas ágiles para el desarrollo de software: XP, Scrum, etc. En este sentido, es evidente lo que Yu [1] y Smith [2] defienden sobre el hecho de que la necesidad de innovación en los equipos de software no está tan enfatizada en el producto software en sí, sino en la gestión de proyectos software, para lo que el modelado y la simulación, así como la gamificación [3], no solo son útiles sino necesarios.

El consumo de recursos en la aplicación de los procesos de ingeniería del software conlleva gastos que, dependiendo de cada proyecto, pueden ser de significativa importancia, por lo que estrategias que sirvan para reducir, prevenir o controlar el consumo excesivo de recursos, son muy valoradas y tienden a tener una gran receptividad en las organizaciones. Según Kellner et al. [4] el modelado y simulación de procesos de desarrollo de software ha ido ganando

el interés de los investigadores y desarrolladores dado que se convierte en una herramienta para analizar la complejidad del negocio del proceso software como un todo y para buscar respuesta a todas las interrogantes que se presentan en este proceso. Además, hay al menos dos poderosas razones para trabajar con modelos de procesos software: conseguir mejores formas de definir y guiar el desarrollo, mantenimiento y evolución del proceso; y, conseguir mejores formas de mejorar los procesos al nivel de actividades y del proceso como un todo [5].

El enfoque del modelado y la simulación ha estado orientado hacia procesos industriales, químicos, tangibles, medibles, cuyos parámetros están bajo el control absoluto de los implicados en el proceso. Sin embargo, en los últimos años se ha dado cierta evolución en el espectro de enfoques, motivada por la necesidad de acceso al control de aspectos como la gestión estratégica [6], la búsqueda de mejoras de los procesos, o el entrenamiento en cuanto a gestión de proyectos de software, desde una perspectiva cercana a los datos reales y con garantía de replicabilidad práctica [7].

Un posible impacto importante, que motiva a perseguir avances en el modelado y simulación de procesos software, es la reducción de la brecha que hay entre el equipo de desarrollo y las personas interesadas o involucradas como beneficiarios del producto a desarrollar. Una herramienta de simulación del proceso software puede representar un lenguaje común que permite el acercamiento entre desarrolladores y *stakeholders,* capaz de habilitar un entorno de diálogo en torno al proceso [8]. Esto es de interés para los desarrolladores de software, que cuentan con representaciones técnicas, como UML (entre otras muchas [9]), para abstraer y representar su sistema a desarrollar, las cuales pueden resultar confusas para los *stakeholders*, que suelen dominar un lenguaje mucho menos técnico y más cercano al mundo de la empresa, pudiendo perder interacción y comprensión suficiente del dominio del problema, si éste es abordado en conjunto con el equipo de desarrollo [10], [11].

Los avances tecnológicos surgen y se desarrollan a un ritmo bastante acelerado y en paralelo a la disponibilidad de equipos potentes de cómputo surgen exigencias cada vez mayores de rendimiento por parte de los equipos humanos de desarrollo de software, así como de contar con estrategias y herramientas que permitan estimar el desempeño de los equipos de producción y los procesos que ellos implican. Debido a esto es menester contar con sistemas simulación y predicción de procesos software que apoyen y respalden la toma de decisiones en torno a todo el proceso de desarrollo. Para ahondar en torno a esto, se puede partir de dos bases de conocimiento concretas que ya cuentan con numerosos trabajos de investigación y proyectos y desarrollos concretos, lo que se detallará a continuación.

Por un lado, existe un marco conceptual claramente estructurado para el modelado y simulación de sistemas en general (Simulation Modelling), con múltiples aplicaciones y usos en áreas como la industrial, ambiental, química, electromecánica, etc. [6]; y por otro lado, se cuenta con una base conceptual en torno a la comprensión, análisis, diseño y monitorización de procesos de desarrollo de software; lo que permite explorar diferentes modelos de proceso, desde

los más clásicos como el modelo estructurado en cascada (Waterfall), hasta modelos orgánicos y menos rígidos como el Rational Unified Process (RUP) [12], o el que se explica más adelante en este trabajo, el Método de Craig Larman [13].

A partir de las dos bases de conocimiento mencionadas anteriormente, surge un interés de investigación en torno al modelado y simulación de procesos software, campo hasta el momento poco explorado y en el que la diversidad y dinamismo de los factores intervinientes aporta un grado de complejidad que afecta notablemente el comportamiento de los equipos de desarrollo y por ende todo el comportamiento organizativo. Se presenta aquí un trabajo en el que se tomó el Método de Larman y se diseñó un modelo de simulación para representar el funcionamiento de dicho método como proceso organizativo de desarrollo de software, dando continuidad al trabajo de Dugarte-Peña [14][8].

El creciente auge en torno al desarrollo de software, las exigencias cada vez más complejas sobre las capacidades de los productos software y el deseo, cada vez mayor, por parte de la industria y de la academia, de comprender y acercar los productos software a la realidad a la que sirven; hacen que sea una necesidad importante contar con herramientas para comprender estos sistemas y su complejidad. El modelado y simulación de los procesos de desarrollo de software representa una oportunidad importante para ambos colectivos en el sentido de que con un mínimo consumo de recursos permite hacer exploraciones sobre el sistema emulado como si se tratase del sistema real, apoyando la toma de decisiones y proyectando escenarios que en la mayoría de los casos son inexplorables o inaccesibles por los altos costes que suponen.

El Método de Larman, ilustrado en la Figura 1, al ser iterativo, incremental y dirigido por casos de uso, ha sido escogido en este trabajo como modelo ideal del proceso a ser simulado. Para hacer el modelado y simulación de este sistema "real"[1], se ha decidido trabajar con el enfoque de la Dinámica de Sistemas por la amplia documentación existente sobre su uso y aprovechamiento y por el valor agregado que le otorga el carácter sistémico que inspiró su nacimiento y las herramientas existentes para su uso. Además, se justifica su uso por la aún vigente importancia de los estudio *in silico* en el ámbito de la ingeniería del software[15].

La Dinámica de Sistemas proporciona todo el marco conceptual para la construcción del modelo. Se ha decidido trabajar con VENSIM [16] por ser una herramienta con amplia documentación a disposición de la comunidad, con una gran colección de experiencias de uso que argumentan su fiabilidad y por ofrecer a la comunidad académica una versión gratuita para fines académicos, como es el caso de este trabajo. Además de permitir la representación del sistema real,

[1] Se usa la palabra "real" para diferenciar del sistema simulado, siguiendo la metodología de Modelado y Simulación de Sistemas. EL sistema "real" hace referencia a la situación compleja de interés, que en este caso es el proceso complejo de desarrollo, teniendo en cuenta todos sus factores, interrelaciones y perspectivas. El sistema simulado, en cambio, será una abstracción restringida de esta "realidad compleja".

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

VENSIM es fácilmente manipulable y modificable para experimentar con variaciones de los parámetros usados en el modelo, permitiendo así modificaciones del estado del sistema que se pueden observar, medir y repetir, resultando idóneo para modelado y la simulación de sistemas, no sólo en el campo de la Ingeniería del Software sino en general [17].

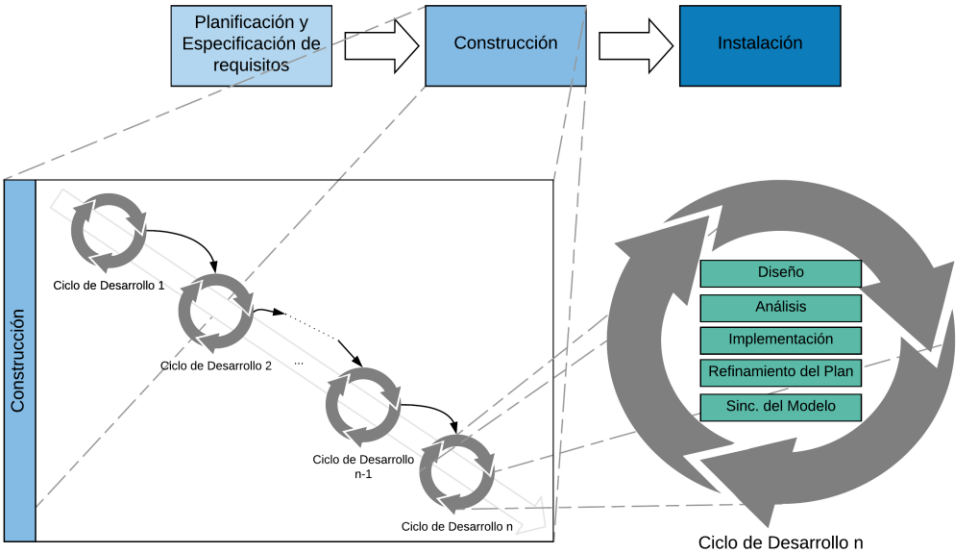

Figura 1. Bosquejo conceptual del método de Larman.

Selección de Parámetros del Proceso Software

En este modelo de simulación del proceso software se hará uso de los parámetros del proceso software que el modelo COCOMO ofrece. COCOMO® II (*Constructive Cost Model II*), es una versión más completa y mejorada de la herramienta COCOMO® (*Constructive Cost Model*), consistente de la implementación de un modelo de estimación implementado por Barry Boehm a finales de los años setenta, en el seno de la University of Southern California [18]. COCOMO® II permite la estimación de coste, esfuerzo y programación de tareas en el proceso de planificación de un nuevo desarrollo de software. Además, permite la manipulación de tres sub-modelos: el de Composición de Aplicaciones (*Applications Composition*), el de Diseño Temprano (*Early Design*), y el de Modelos Post-arquitectura (*Post-architecture models*). Con respecto a los aspectos de un proyecto de desarrollo de software, Cocomo II se fija en cuatro tipos de atributos de un proceso software para hacer la estimación. Una descripción detallada de estos atributos, las funciones de estimación que usa y como se interrelacionan los atributos entre sí, se encuentra disponible en [19]. Debido a su importancia para este trabajo, el listado de atributos por tipo se presenta en la Tabla 1. Se muestra también

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

una escala numeral que aporta los valores referenciales de entrada a los modelos y fórmulas matemáticas de las que se hace uso para estimar el coste y esfuerzo de un proyecto.

Tabla 1. Parámetros-Atributos del modelo de Estimación de costes COCOMO II.

COCOMO II: Parámetros y atributos		Valor					
Nombre	**Atributo**	**Muy bajo**	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra alto**
Atributos de Software							
RELY	Fiabilidad	0.75	0.88	1.00	1.15	1.40	__
DATA	Tamaño de Base de Datos	__	0.94	1.00	1.08	1.16	__
CPLX	Complejidad	0.70	0.85	1.00	1.15	1.30	1.65
Atributos de Hardware							
TIME	Restricciones de tiempo de ejecución	__	__	1.00	1.11	1.30	1.66
STOR	Restricciones de memoria virtual	__	__	1.00	1.06	1.21	1.56
VIRT	Volatilidad de la máquina virtual	__	0.87	1.00	1.15	1.30	__
TURN	Tiempo de respuesta	__	0.87	1.00	1.07	1.15	__
Atributos de Personal							
ACAP	Capacidad de análisis	1.46	1.19	1.00	0.86	0.71	
AEXP	Experiencia en la aplicación	1.29	1.13	1.00	0.91	0.82	
PCAP	Calidad de los programadores	1.42	1.17	1.00	0.86	0.70	__
VEXP	Experiencia en la máquina virtual	1.21	1.10	1.00	0.90	__	__
LEXP	Experiencia en el lenguaje	1.14	1.07	1.00	0.95	__	__
Atributos del Proyecto							
MODP	Técnicas actualizadas de programación	1.24	1.10	1.00	0.91	0.82	__
TOOL	Utilización de herramientas de software	1.24	1.10	1.00	0.91	0.83	__
SCED	Restricciones de tiempos de desarrollo	1.22	1.08	1.00	1.04	1.10	__

Modelado y Simulación del Proceso Software

Teniendo como referencia la metodología de [20] para modelado con Dinámica de Sistemas, y la utilidad que tiene para ser explotada en el ámbito de la ingeniería del software[21]–[23], primero se hizo una descripción del sistema a modelar, después una representación de las causalidades existentes entre los elementos del sistema, seguido por la identificación y representación de los elementos propios de la Dinámica de Sistemas, y finalmente la configuración del modelo para su posterior uso, evaluación, y análisis de funcionalidad.

El sistema que se va a modelar está compuesto por un conjunto de elementos intervinientes en el proceso de desarrollo de software como lo son: personas (analistas, programadores, directores, etc.), equipos informáticos, requisitos, activos de proceso software (código, especificaciones, prototipos, documentación), y elementos propios del entorno del sistema

como las restricciones o fronteras (tiempos, capacidades máximas, etc.). Todos estos elementos están interrelacionados de alguna manera en alguna o en todas las fases que comprende el proceso de desarrollo de software, sin embargo, el funcionamiento de este conjunto de elementos sobrepasa esa mera descripción, resultando realmente en un comportamiento dinámico y lleno de causalidades complejas y variantes en el tiempo.

El Método de Craig Larman se compone de tres fases o macro etapas claramente diferenciadas una de otra, y en esencia comunes a la gran mayoría de modelos y métodos de desarrollo de software: Planificación y Especificación de requisitos, Construcción e Instalación.

Diagrama causal del modelo.

A continuación, como si se tratase de una especie de "zoom" o "acercamiento" hacia lo que realmente se quiere representar, se muestra primero un macro diagrama que muestra cómo se relacionan las tres fases del proceso de desarrollo ya explicadas: planificación y especificación de requisitos, construcción e instalación.

Seguidamente se muestra un primer grado de dinamismo e interactividad que se da en el proceso de software consecuencia de que en la fase de construcción están implicados no uno sino varios ciclos de desarrollo que son necesarios para alcanzar el grado de madurez aceptable del producto software que se desea construir.

Finalmente se presenta un diagrama causal detallado del sub-proceso de la fase de construcción, lo que es un precedente y requisito para construir el modelo de Dinámica de Sistemas, es decir, el modelo en términos de niveles y flujos dinámicos.

Macro diagrama del Proceso General de Desarrollo de Software de Craig Larman.

Las tres fases del proceso están interconectadas, y, como se observa en las Figuras 2 y 3, suelen tener una fuerte interacción durante todo el proceso de desarrollo. Aunque la fase de planificación y especificación de requisitos es la inicial, desde las fases de construcción y de instalación se hacen constantes referencias a esta fase para realizar ajustes que corresponden a actividades de planificación y especificación de requisitos. Lo mismo ocurre entre las fases de Construcción e Instalación, lo que significaría que durante la instalación pueden generarse solicitudes de "re-engineering" de actividades que corresponden a la fase de construcción, es decir, se hacen cambios en los requisitos. Sin embargo, el núcleo de todo el proceso, o la fase en que la carga de trabajo es significativamente mayor, es la fase de construcción, que implica fuerte consumo de recursos para lograr obtener el producto listo para implantación a partir de su especificación.

Macro diagrama de la fase "Construcción" de un proceso de desarrollo de software.

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

La fase de Construcción de un proceso de desarrollo de software tiene por sí misma cierta complejidad y comportamiento dinámico que merece ser estudiado y comprendido, tanto desde un punto de vista sistémico y sinérgico, como comprendiendo a su vez las distintas subetapas que contiene. Estas subetapas son los llamados ciclos de desarrollo que se llevan a cabo formalmente con posterioridad a la Planificación y Especificación de Requisitos y derivan en la presentación de un producto software que luego estará disponible para ser llevado a la fase de instalación.

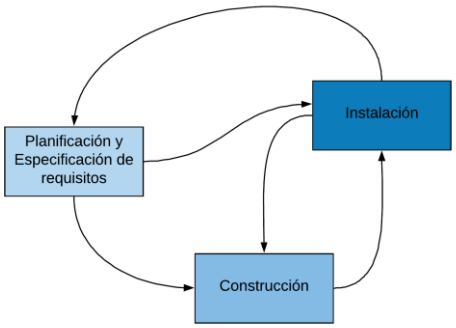

Figura 2. Diagrama general de fases del Método de Larman.

Como se puede ver en la Figura 3, los ciclos de desarrollo están relacionados entre sí. En cada ciclo de desarrollo se van atendiendo casos de uso de los que se definen en la planificación y especificación de requisitos. El orden en que estos casos de uso son atendidos responde a la priorización que se hace sobre ellos, y, además, un caso de uso puede ser parte de más de un ciclo de desarrollo si la complejidad del caso lo amerita.

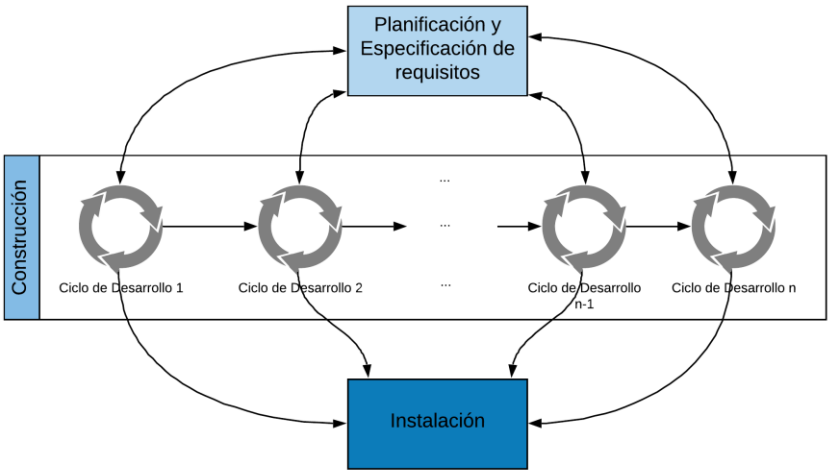

Figura 3. Primer desdoblamiento de la complejidad del proceso: Construcción.

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

La cantidad de ciclos de desarrollo no es exacta, sino que se determina para cada caso en particular según la complejidad del proyecto y de sus casos de uso. En la Figura 3, observamos un primer "acercamiento" o "zoom" sobre el proceso general de desarrollo, en el que se muestra que la fase de construcción implica a su vez varios ciclos de desarrollo y que estos no son independientes uno del otro sino que están relacionado y hay relaciones de causalidad entre ellos: Para el ejemplo de la Figura 3, el ciclo 1 produce efectos en ciclos 2 y 3, y el ciclo 2 que se ha visto influenciado por el ciclo 1, también tendrá relevancia sobre el ciclo 3.

Diagrama en detalle: Fase de construcción

A continuación, en la Figura 4, se presentará el diagrama causal del sistema real que se está modelando en este trabajo. En este diagrama, a forma de segundo "acercamiento" o "zoom" sobre el sistema real, se han incorporado todos los elementos que se considera tienen algún efecto en la fase de construcción del proceso software. Se ha agregado la mayor cantidad de elementos posibles con la finalidad de hacer la representación pictográfica más representativa.

Siguiendo el sentido de las flechas, se puede apreciar relaciones de causalidad entre los elementos a los extremos de las flechas, lo que nos permite apreciar la complejidad del proceso software al haber no sólo una cantidad considerable de flechas sino también un número elevado de realimentaciones, lo que según [20] justifica el uso de la Dinámica de Sistemas para modelar sistemas como éste. A continuación, se presentan los elementos clasificados en términos de la dinámica de sistemas (Detalles se han omitido en este artículo, pero las definiciones extensas se alinean con [13], [24]).

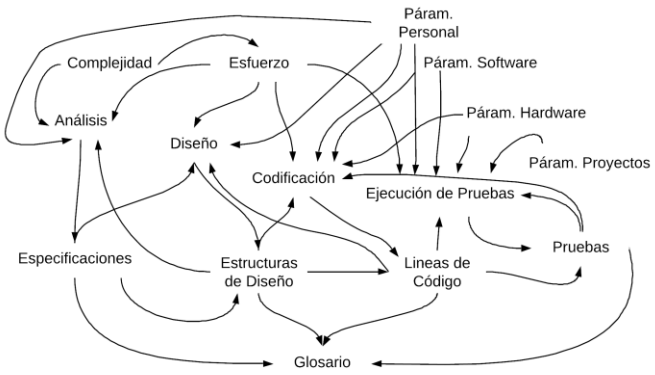

Figura 4. Diagrama causal de la fase "Construcción" del Método Larman de ingeniería del Software.

Identificación de Acumuladores, Niveles o Stocks

Para el proceso de Ingeniería del Software aquí presentado, los elementos que tienden a acumularse en el tiempo y que por ende representan "niveles del sistema" en el dominio de la Dinámica de Sistemas son: Especificaciones, Estructuras de Diseño, Código, Pruebas realizadas y un nivel cuya funcionalidad es transversal pero no crítica para todo el proceso: Glosario.

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

Identificación de flujos o tasas de crecimiento y decrecimiento.

A continuación, se presentan los elementos del sistema que representan de alguna manera flujos de crecimiento y decrecimiento de los niveles anteriormente especificados: Análisis, Diseño, Implementación, Corrección y Pruebas.

Modelo computacional

El modelo que se presenta, debido a su tamaño, comprende tres vistas de modelo, tomando ventaja de la propiedad de modelado paralelo del software Vensim. Inicialmente se pretendía construir un modelo general que englobara las tres fases del Método de Larman, sin embargo, durante el proceso de modelado fue necesario delimitar el proceso software de manera que las fases de "Planificación y Especificación de Requisitos" e "Instalación" sólo se llevan a cabo una vez, e implican tareas concretas, mientras que la fase de "Construcción" tiene implícita toda una dinámica (e iteratividad) en sí misma. La Figura 5 muestra el submodelo de la fase de *Planificación y Especificación de Requisitos*.

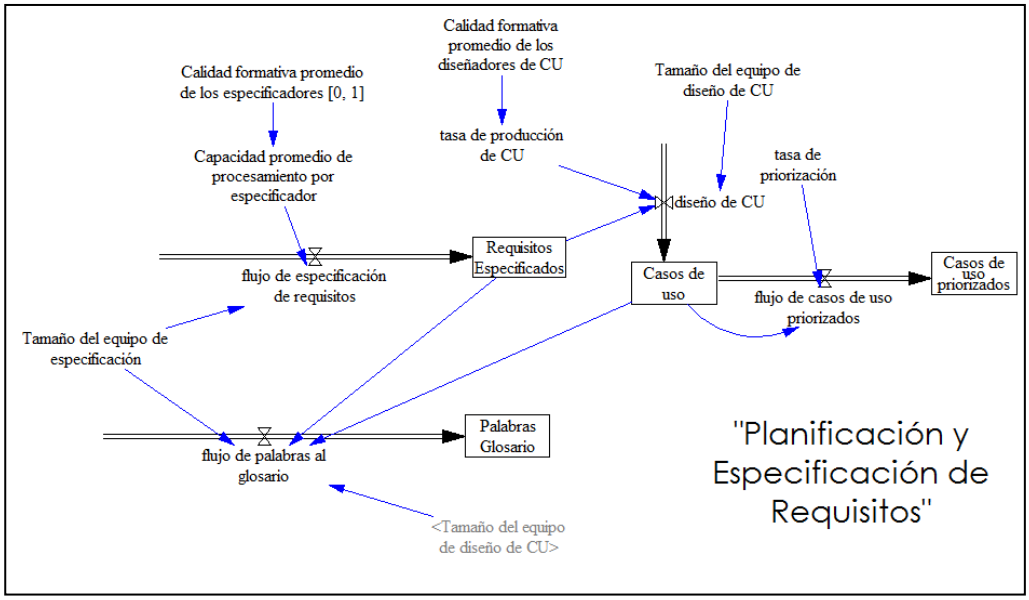

Figura 5. Submodelo de simulación para la fase "Planificación y Especificación de Requisitos".

Seguidamente se presenta el modelo de simulación para la fase de construcción (en tres vistas, o niveles de abstracción que forman parte del mismo modelo). La primera vista, presenta de forma esquemática los parámetros del proceso que afectan el proceso software que fueron explicados anteriormente. Estos parámetros tienen efecto en distintas partes del modelo de simulación, sin embargo, se han conglomerado en esta primera vista para facilitar su manipulación y la fácil calibración del modelo al momento de hacer pruebas variando sus valores. En esta primera vista (Figura 6) también se ha incorporado la parte del modelo en la que se implementan las ecuaciones usadas por [19] para la estimación de esfuerzo.

Revista Innovación y Software
Vol. 1, No. 1, Mes Marzo-Agosto, 2020
ISSN: 2708-0935
Pág. 39-57
https://revistas.ulasalle.edu.pe/innosoft

La segunda vista del modelo en VENSIM (Figura 7) es la parte central del modelo, y en ella se han agregado los elementos principales representativos del proceso de desarrollo de software.

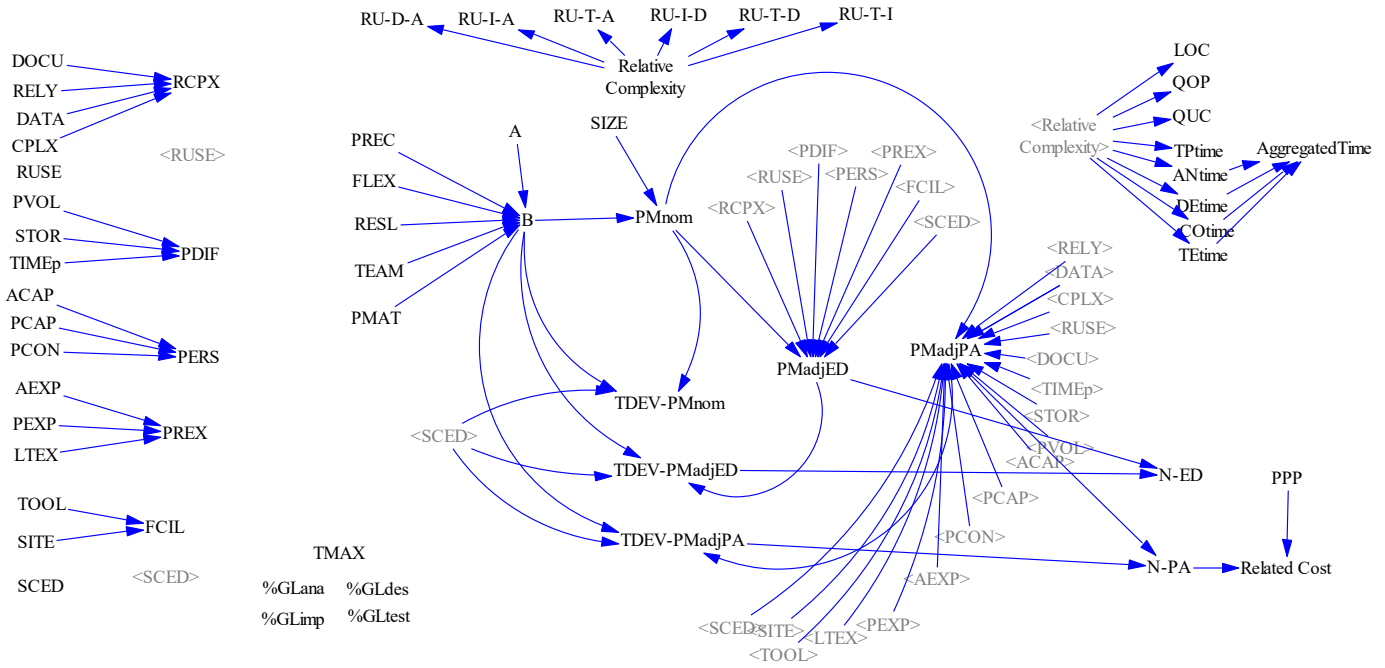

Figura 6. Vista 1: Submodelo de configuración de parámetros y estimación.

En la Figura 6 se pueden observar los elementos del modelo COCOMO II de estimación, y como éstos afectan la estimación tanto en el modelo “Early Design” (ED) como “Post Architecture” (PA). Se han respetado los acrónimos del modelo COCOMO tanto para los atributos del modelo (Variables auxiliares en esta vista) como para las variables producto de la estimación (Estimaciones ED y PA).

La Figura 7 muestra la vista del modelo que implementa las los elementos del proceso de desarrollo de software y cómo estas se van acumulando a medida que el tiempo de proyecto transcurre. Las fases que se “acumulan”, es decir, que se van cumpliendo durante el proceso son: Analysis, Design, Code, y Test. Hay otro nivel que se acumula referente a la incorporación de términos al Glosario de un proyecto. Los flujos que nutren estos niveles determinan el ritmo con el que crecen o decrecen y son dependientes de distintos factores que afectan el proceso, que en este caso se tomaron del modelo COCOMO de estimación, como se ha explicado anteriormente.

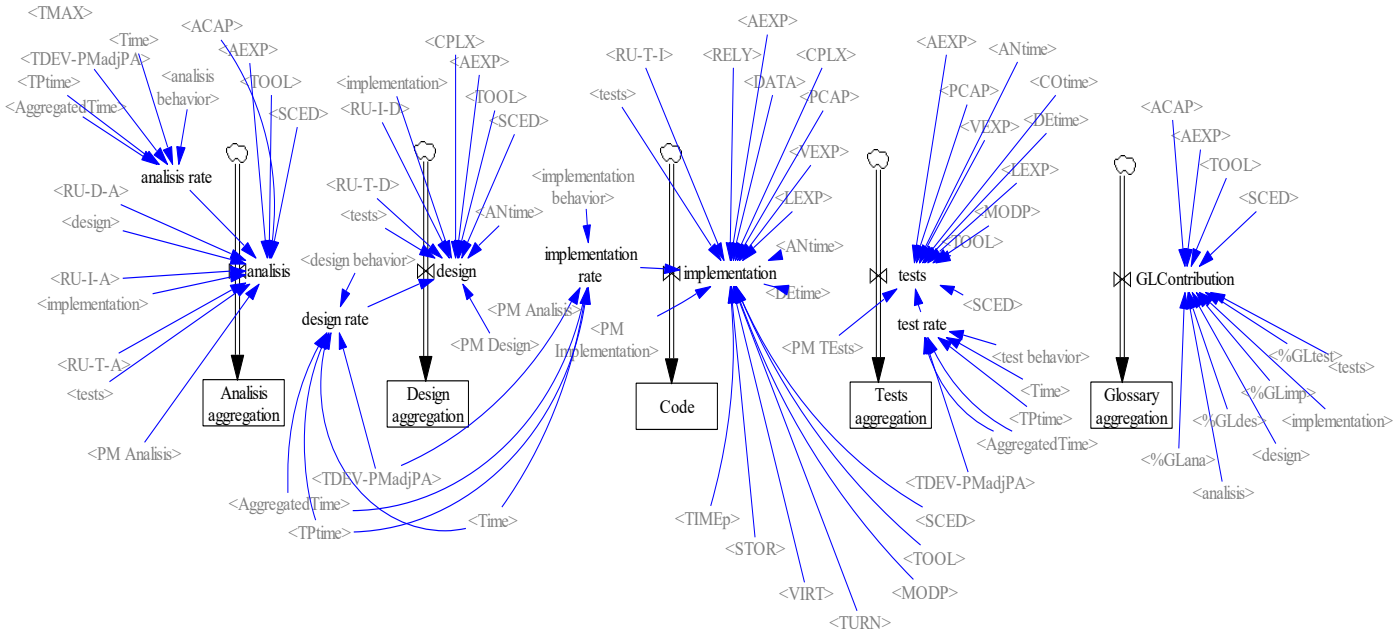

Figura 7. Vista 2: Submodelo de la fase de "Construcción" en un proceso de desarrollo.

La tercera vista (Figura 8) se ha agregado para representar el consumo paulatino del esfuerzo correspondiente a cada una de las etapas de la fase de construcción de software. Se puede observar que el esfuerzo distribuido está acumulado en los niveles PM Analysis, PM Design, PM Implementation, y PM Tests. Los flujos que salen de cada nivel definen el ritmo en que este esfuerzo es consumido, y como se puede observar, son afectados por distintas variables del proceso.

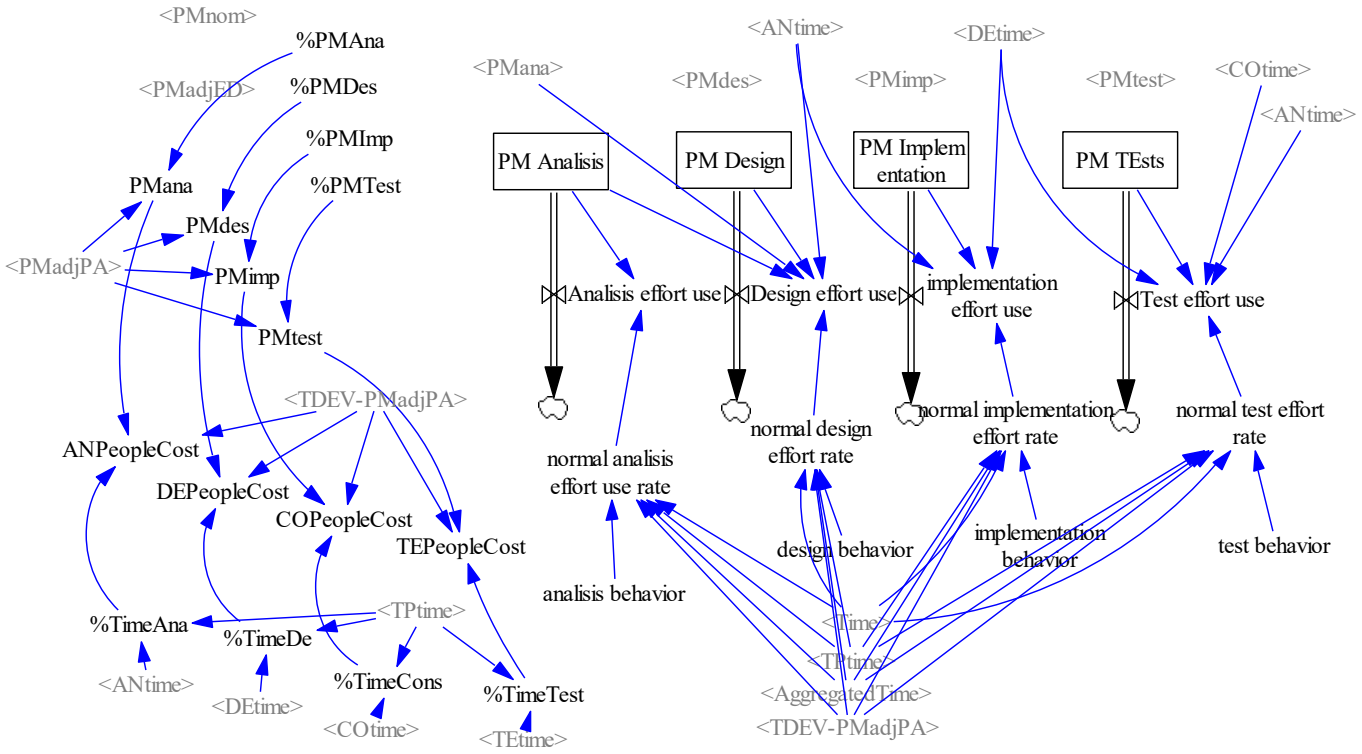

Figura 8. Vista 3: Submodelo de distribución del esfuerzo en el proceso de desarrollo de software

Ejecución y discusión de la simulación del modelo computacional

Se ejecutó el modelo con los parámetros de COCOMO en sus valores "nominales", obteniéndose lo que se esperaría sea el comportamiento normal del sistema simulado. Se puede observar una gran cantidad de factores y su variación en el tiempo; sin embargo, se enfocará la atención en mostrar aquellos aspectos del proceso que son de interés en la gestión de proyectos software**: el comportamiento del sistema como un todo, el consumo de los recursos en el transcurso del tiempo, y los comportamientos de los acumuladores por tratarse de "variables de estado" que permiten crear panoramas generales del funcionamiento del sistema**.

En la Figura 8, se observa el comportamiento general del sistema con la figuración de parámetros nominales. Se puede observar como la realización de las actividades del proceso de construcción (representadas en el eje y como unidades atómicas de tareas por cada fase), aunque con ciertas simultaneidades, se concentra en su mayoría en distintas etapas (temporales) del proceso general. El análisis y el diseño tienden a concentrarse en la primera mitad del ciclo de vida del proceso, mientras que la implementación (y su correspondiente construcción de código fuente) se concentra en la mitad del proceso y las pruebas hacia la segunda mitad del ciclo de vida del proceso. Este comportamiento presenta cierta

similitud con el proceso de software descrito en el *Rational Unified Process*, lo que da una primera idea de la aproximación de este modelo de simulación a los modelos teóricos existentes en la literatura.

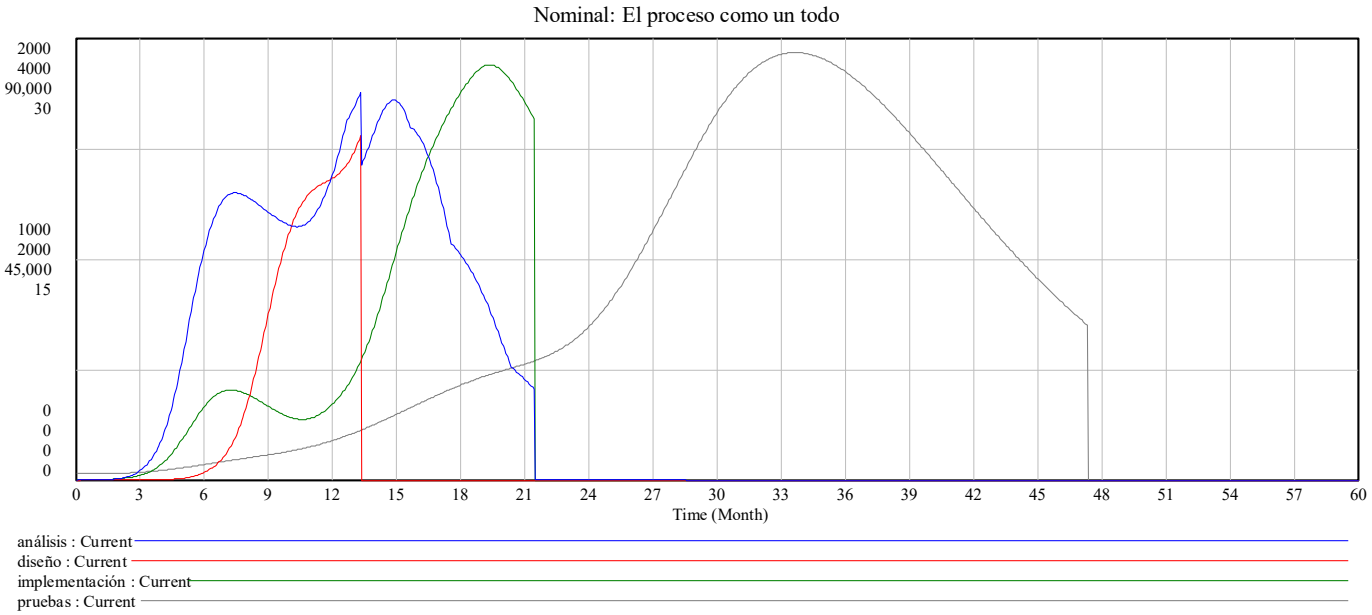

Figura 8. Las fases de la "construcción" del proceso software: modelo nominal

Si se piensa en las actividades del proceso de construcción como metas que deben ser alcanzadas en el transcurso del tiempo, la Figura 9, es una ilustración del comportamiento de la maduración del alcance de estas metas. Se puede ver que los primeros conjuntos de actividades en completarse son los correspondientes a “Análisis” y “Diseño”, seguido por el correspondiente a la construcción del código, y finalmente las actividades de pruebas, que al principio del proceso eran muy pocas, pero que dominan el proceso hacia el final, cuando el producto se prepara para ser entregado.

Revista Innovación y Software
Vol. x, No. x, Mes x-y, 2019
ISSN: XXX-XXXX
Pág. x-y
https://revistas.ulasalle.edu.pe/innosoft

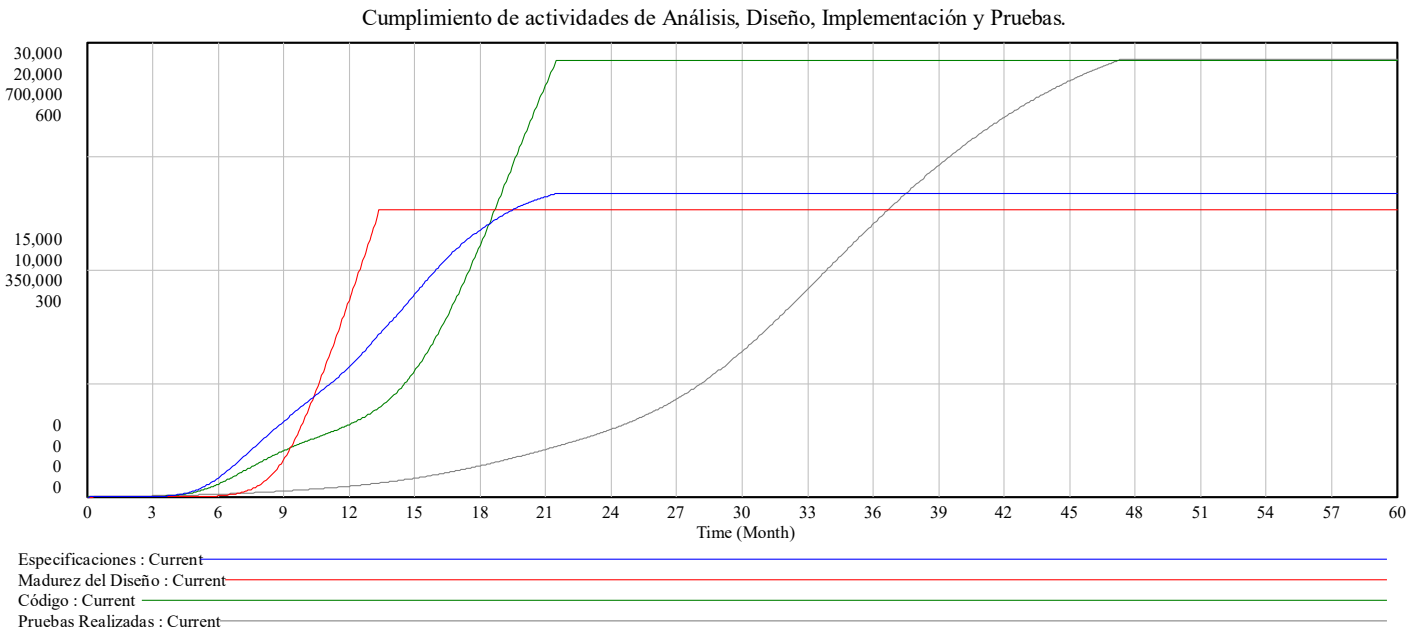

Figura 9. Maduración de las fases del proceso de construcción: modelo nominal.

Una forma también interesante de verificar el funcionamiento del proceso es observando el comportamiento del "esfuerzo", dado que este se distribuye durante todo el proceso en los distintos tipos de actividades que se llevan a cabo. En la Figura 10, (sin escalar) se ve que el esfuerzo consumido en el proyecto se distribuye entre análisis, diseño, codificación y pruebas respectivamente. En dicha imagen se observa en qué momentos de todo el proceso hay mayores flujos de esfuerzo en el proyecto, sin embargo, nótese que como es de esperar, las escalas no se corresponden; ya que,

en cuanto a cantidades, el esfuerzo dedicado a algunas actividades de diseño, es mucho mayor que el dedicado a las pruebas, por ejemplo.

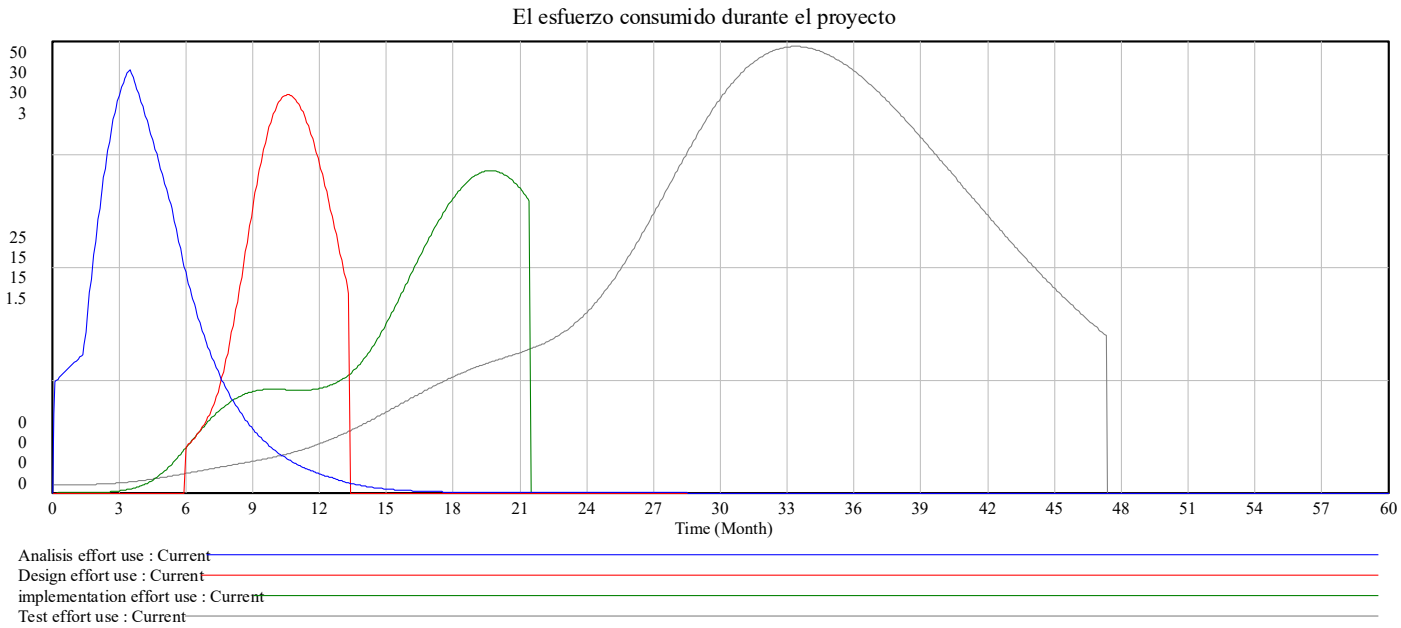

Figura 10. Esfuerzo consumido en el proceso de construcción, separado por fases: modelo nominal.

Adicionalmente, el modelo ha permitido a los autores crear escenarios con los que estudiar el comportamiento del sistema y cómo este se ve afectado al variar los elementos de estimación que afectan el proceso. Los resultados parciales se presentan en [8] y se refinan para un futuro trabajo enfocado en el efecto que tienen los parámetros de estimación del producto software en el desempeño del proceso.

Conclusiones

En este trabajo se decidió trabajar con la Dinámica de Sistemas como enfoque para la construcción del modelo de proceso software, basados en el hecho de que el proceso software, lejos de ser metódico y procedimental resulta ser complejo, variante y en muchas ocasiones contraintuitivo y con muchos fenómenos emergentes e impredecibles.

Para la construcción del modelo, y atendiendo a la metodología tomada como referencia [20]; se hizo primero una profunda inmersión en el sistema a modelar, lo que se tradujo en una comprensión profunda y sistémica del proceso de ingeniería de software del Método de Larman, su funcionamiento, fases, características, y propiedades, constituyendo lo que es un *primer nivel de abstracción del proceso*. Parte de esta inmersión fue la exploración de enfoques que describieran los parámetros que afectan el proceso, para lo que resultó de gran utilidad y aprovechamiento hacer uso del modelo de costes COCOMO II, que tiene definidos una serie de parámetros comunes al proceso que tienen

incidencia en el desarrollo y en el consumo de recursos relacionado. En el modelo que aquí se presentó se hizo uso de estos parámetros como una forma de incrementar la representatividad del modelo, lo que permitió incorporar los efectos de los parámetros del proceso, sobre el funcionamiento del proceso como un todo, y de esta manera estimar posibles efectos y comportamientos esperados a partir de ciertas configuraciones de parámetros; representando esta propiedad un aporte significativo para quienes quieren estudiar el proceso software con mínimos costes y por medio de una manera práctica, manejable, utilizable por múltiples usuarios y con un importante potencial para dar soporte a la toma de decisiones en los equipos de gestión de procesos software.

Lo anterior fue el fundamento para la construcción de los diagramas causales representativos, inicialmente de todo el proceso, y después, *a manera de un zoom dentro del sistema general*, de la fase de *construcción* del proceso de ingeniería de software. El aporte principal en este aspecto vino al momento de modelizar dicho sistema (el subproceso de construcción) haciendo uso del enfoque de la Dinámica de Sistemas. En esta modelización, la identificación de variables, niveles y flujos que interactúan en el sistema, representan un *segundo nivel de abstracción del sistema,* de suma importancia y utilidad, ya que aquí se identifican relaciones entre los elementos del sistema que no siempre son evidentes a simple vista ni con una exploración superficial. Posteriormente, todas las relaciones, causalidades y ciclos de realimentación identificados fueron implementados en el diseño, configuración, validación y manipulación del modelo en sí, logrando un nivel de representatividad del sistema bastante amplio y conformando una de las principales fortalezas de este trabajo, al ser **un modelo representativo del proceso software, que toma en cuenta un gran número de elementos y parámetros de distinto tipo que inevitablemente tienen influencia sobre el sistema y que no se deben ignorar**.

También han surgido argumentos para sustentar la afirmación de que **es necesario ahondar en lograr contar con un enfoque metodológico estructurado que sirva de contexto para estudiar problemas similares al que aquí se enfrentó: simular procesos de ingeniería del software**. Se trataría de un enfoque en el que los criterios para escoger un enfoque u otro sean un activo a disposición del usuario o investigador, cosa que permitiría múltiples beneficios tanto para las ciencias en general como para los campos de ésta que se dedican a comprender los fenómenos organizativos, entre los que se cuentan los del proceso software.

Igualmente, es de importancia para investigaciones futuras, la ejecución de experimentos en los que bajo las mismas condiciones y con la misma fijación de parámetros se exploren simulaciones basadas en diferentes modelos de un mismo sistema software “real”. Esto permitiría por un lado contrastar un modelo con otro con respecto a su grado de representación de la complejidad del sistema real, y por el otro permitiría una observación sistémica que permitiría estimar otros aspectos diferenciadores de los modelos y enfoques, como son la funcionalidad, grado de usabilidad y

reproducibilidad de un software; asunto que se presenta como uno de los intereses para evolucionar el cuerpo actual de conocimiento de la Ingeniería del Software [25].

Referencias

[1] W. D. Yu, "A Modeling Approach to Software Cost Estimation," *IEEE J. Sel. Areas Commun.*, vol. 8, pp. 309–314, 1990.

[2] C. Smith, "Improving Service While Controlling Costs," *IEEE Softw.*, vol. March, pp. 95–96, 1991.

[3] C. Alejandro and M. Ruiz, "Coverage of ISO/IEC 12207 Software Lifecycle Process by a Simulation-Based Serious Game," in *SPICE 2016: Software Process Improvement and Capability Determination*, 2016, pp. 56–70, doi: https://doi.org/10.1007/978-3-319-38980-6_5.

[4] M. I. Kellner, R. J. Madachy, and D. M. Raffo, "Software process simulation modeling: Why? What? How?," *J. Syst. Softw.*, vol. 46, no. 2–3, pp. 91–105, 1999, doi: 10.1016/S0164-1212(99)00003-5.

[5] S. T. Acuña and M. I. Sánchez-Segura, *New Trends in Software Process Modeling*. Singapore: World Scientific Publishing Co. Pte. Ltd., 2006.

[6] A. Greasley, *Simulation modelling for business*. Routledge, 2017.

[7] M. I. Lunesu, J. Münch, M. Marchesi, and M. Kuhrmann, "Using simulation for understanding and reproducing distributed software development processes in the cloud," *Inf. Softw. Technol.*, vol. 103, no. July, pp. 226–238, 2018, doi: 10.1016/j.infsof.2018.07.004.

[8] G. L. Dugarte-Peña, "Modelado y Simulación de un Proceso de Desarrollo de Software dirigido por el Método de Craig Larman: Una aplicación de la dinámica de sistemas. (Modelling and Simulation of a Craig Larman Methods' Software Development Process: A System Dynamics Approach," Universidad Carlos III de Madrid, 2015.

[9] J. A. García-García, J. G. Enríquez, and F. J. Domínguez-Mayo, "Characterizing and evaluating the quality of software process modeling language: Comparison of ten representative model-based languages," *Comput. Stand. Interfaces*, vol. 63, no. October 2018, pp. 52–66, 2019, doi: 10.1016/j.csi.2018.11.008.

[10] R. Vicente, "Modelamiento semántico con Dinámica de Sistemas en el proceso de desarrollo de software," *Iber. J. Inf. Syst. Technol.*, no. 10, pp. 19–33, Dec. 2012, doi: 10.4304/risti.10.19-34.

[11] S. Robertson, "Learning from Other Disciplines," 2005.

[12] I. Jacobson, G. Booch, and J. E. Rumbaugh, *The unified software development process*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[13] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative*

Development, 3rd ed. Prentice Hall, 2004.

[14] G.-L. Dugarte-Peña, "Software Engineering under the prism of System Dynamics," in *XXIII Jornadas. Internacionales de Ingeniería de Sistemas. Universidad Católica de Santa María.*, 2016.

[15] G. H. Travassos and M. Barros, "Contributions of In Virtuo and In Silico Experimentes for the Future of Empirical Studies in Software Engineering," *2nd Work. Work. Ser. Empir. Softw. Eng. Futur. Empir. Stud. Softw. Eng.*, no. January, pp. 1–14, 2003.

[16] Ventana Systems, "VENSIM." Ventana Systems, 2011.

[17] J. M. García, *Teoría y ejercicios prácticos de Dinámica de Sistemas*. Independently Published, 2018.

[18] *USC COCOMO II: User´s manual*. Center for Software Engineering, USC, 2003.

[19] B. Boehm, "COCOMO II Model Definition Manual," vol. 87, no. 5 Suppl, p. i, 2012, doi: 10.4269/ajtmh.2012.875suppack.

[20] J. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*. USA: McGraw-Hill, 2000.

[21] B. B. Nicolau De França and G. Horta Travassos, "Reporting guidelines for simulation-based studies in software engineering," *IET Semin. Dig.*, vol. 2012, no. 1, pp. 156–160, 2012, doi: 10.1049/ic.2012.0019.

[22] H. Zhang, B. A. Kitchenham, and D. Pfahl, "Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review," in *Making Globally Distributed Software Development a Success Story*, Q. Wang, D. Pfahl, and D. M. Raffo, Eds. Springer, 2008, pp. 345–356.

[23] H. Zhang, B. Kitchenham, and D. Pfahl, "Software Process Simulation Modeling: Facts, Trends and Directions," *2008 15th Asia-Pacific Softw. Eng. Conf.*, pp. 59–66, 2008, doi: 10.1109/APSEC.2008.50.

[24] X. Ferré-Grau and M.-I. Sanchez-Segura, "Desarrollo Orientado a Objetos con UML," pp. 1–38, 2013.

[25] M.-I. Sanchez-Segura, A. Jordan, F. Medina-Dominguez, and G.-L. Dugarte-Peña, "Software Engineers must speak the Systemic Intangible Process Assets Language," 2016.