UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN CIENCIA Y TECNOLOGÍA INFORMÁTICA

TRABAJO DE FIN DE MÁSTER

MODELADO Y SIMULACIÓN DE UN PROCESO DE DESARROLLO DE SOFTWARE DIRIGIDO POR EL MÉTODO DE CRAIG LARMAN: UNA APLICACIÓN DE LA DINÁMICA DE SISTEMAS

AUTOR:

GERMAN LENIN DUGARTE PEÑA

TUTORAS:

MARÍA ISABEL SÁNCHEZ SEGURA

FUENSANTA MEDINA DOMÍNGUEZ

Leganés, Madrid, Septiembre de 2015.

TRABAJO DE FIN DE MÁSTER

TÍTULO: MODELADO Y SIMULACIÓN DE UN PROCESO DE DESARROLLO DE SOFTWARE DIRIGIDO POR EL MÉTODO DE CRAIG LARMAN: UNA APLICACIÓN DE LA DINÁMICA DE SISTEMAS

AUTOR: GERMAN LENIN DUGARTE PEÑA

TUTORA: MARÍA ISABEL SÁNCHEZ SEGURA

COTUTORA: FUENSANTA MEDINA DOMÍNGUEZ

TRIBUNAL

PRESIDENTE: MARÍA ISABEL SÁNCHEZ SEGURA

VOCAL: SERGIO PASTRANA PORTILLO

SECRETARIO: JOSÉ ANTONIO IGLESIAS MARTÍNEZ

Tras la defensa y presentación de éste Trabajo de Fin de Máster el día 28 de Septiembre de 2015 en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, el tribunal le otorga la siguiente calificación:

CALIFICACIÓN:

Agradecimientos

Quiero agradecer a mi hijo Ramsés Leonardo, que a su corta edad, y sin saberlo, se ha convertido en mi más preciada e inagotable fuente de maná de vida…

A mi esposa, a quien amo y es el pilar del trípode que en familia constituimos…

A mi querida familia en Venezuela, que a pesar de la distancia, hacen de todo por demostrarme su incondicional apoyo…

A las profesoras Maribel y Fuensanta que han tenido una paciencia inagotable en la labor de apoyarme y orientarme en la realización de este trabajo…

A FCC-Fomento de Construcciones y Contratas, y a mis compañeros de trabajo allí, que me han brindado su apoyo, consejos e instalaciones para desarrollar esta memoria…

A todo el que se interese en leer este trabajo, que de alguna manera estará en el mismo mundo paralelo que yo…

Resumen

El proceso software involucra costes y consumos de recursos muy importantes, de los que no se puede disponer para simple experimentación; por lo que es de gran importancia contar con herramientas que permitan incrementar el marco decisorio con información y criterios sobre posibles comportamientos del proceso ante distintos escenarios.

El presente trabajo trata del modelado y simulación de un proceso de desarrollo de software. El enfoque usado para la construcción del modelo de simulación es la Dinámica de Sistemas, que permite representar gráficamente los elementos que intervienen en el proceso software, diferenciando aquellos que son variables, auxiliares, niveles, flujos o relaciones; e incorporando por medio de estos elementos la mayor cantidad posible de parámetros que afectan el proceso. Para la incorporación de los parámetros, se tomó como referencia los que usa el modelo de estimación de costes COCOMO, que ya cuenta con una fundamentación teórico-práctica que avala su fiabilidad. Para la construcción del modelo, se tomó como referencia del *sistema real*, el modelo de proceso software definido por Larman, más conocido como "Método de Larman".

El modelo de Dinámica de Sistemas presentado, permite hacer estimaciones iniciales del comportamiento del proceso software y de los elementos que le conforman, durante el transcurso de un tiempo de simulación configurable. Esto es posible por medio de la observación y el estudio de las variables de estado del sistema, que permiten concluir sobre efectos de los parámetros del sistema en el comportamiento del sistema en general, y por ende llevar a cabo estudios de escenarios. El modelo deriva en una potencial herramienta de soporte a los equipos de gestión de proyectos software, y a las empresas que hacen de la Gestión de Proyectos su negocio principal, por lo que se abre una variedad de caminos de investigación en torno a esta área de conocimiento.

Palabras clave:

Modelado y Simulación de Procesos Software, Gestión del Proceso Software, Modelado y Simulación de Sistemas, Método de Craig Larman, Dinámica de Sistemas

Abstract

The software process involves very important costs and resources waste, not simply available for experimentation; which is why it is very important to have at our disposal some tools able to allow improving the decision framework with information and criteria about possible and expected behaviour of the software process against different scenario or parameters configurations.

This work is an attempt to deal with the task of modelling and simulating a software development process. The chosen approach for designing and building the simulation model is System Dynamics, an approach that enable users to represent graphically all elements that take part into the software process, while allowing differentiation between variables, auxiliaries, stocks, flows or relations, and, by using them, the incorporation to the model of many relevant elements as possible. In order to choose the right parameters to be added to the model, the COCOMO estimation model was used; which has a predefined set of parameters that represent the software process behaviour drivers. As an ideal, reality-based, software process conceptual model, the Craig Larman Software Process model was chosen.

The System Dynamics model developed here, allows one to make some initial estimation of the software process and its elements behaviour in the course of the time. This estimation and knowledge-emerging work is possible thanks to the observation and study of the system's state variables, empowering one to discern about the effect that changes in the parameters produce to the general process, hence, carrying out relevant scenario studies. This model becomes a potential tool for supporting Software-development Project Managers and enterprises dedicated to Software-Projects Management, meaning the starting point for future research from this point and on.

Keywords:

Software Process Simulation Modelling, Software Process Management, Systems Modelling and Simulation, Craig Larman`s Method, Systems Dynamics

Índice General

Lista de Figuras

Lista de Tablas

1 Introducción

1.1 Motivación

Con el acelerado crecimiento de las capacidades de procesamiento de información, también han ido creciendo aceleradamente los ritmos con que las fábricas de software deben completar los procesos para desarrollar sus productos. A la vez que es posible trabajar con máquinas cada vez más potentes, es posible asumir proyectos de mayores dimensiones, por lo que también se incrementa la complejidad de los proyectos y sus requisitos de gestión.

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. 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, 1990) y (Smith, 1991) 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 simulación no solo es útil sino necesario.

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, Madachy, & Raffo, 1999) 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.

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, la búsqueda de mejoras de los procesos, o el entrenamiento en cuanto a gestión de proyectos de software.

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 de acercamiento entre desarrolladores y *stakeholders* capaz de habilitar un entorno de diálogo en torno al proceso en sí. Mientras que los desarrolladores de software cuentan con representaciones técnicas, como UML, para abstraer y representar su sistema a desarrollar, éstas representaciones pueden resultar confusas para los *stakeholders*, que suelen dominar un lenguaje mucho menos técnico y más cercano al mundo de la empresa, por lo que pueden perder interacción y comprensión suficiente del dominio del problema, si éste se aborda en conjunto con el equipo de desarrollo. (Vicente, 2012a), (Robertson, 2005).

1.2 Definición del problema

A partir del hecho de que los avances tecnológicos surgen y se desarrollan a un ritmo bastante acelerado y de que en paralelo a la disponibilidad de equipos potentes de cómputo surgen las exigencias cada vez mayores de rendimiento por parte de los equipos humanos de desarrollo de software, surge la necesidad de contar con estrategias

y herramientas que permitan estimar el desempeño de los equipos de producción y los procesos que ellos implican, por lo que es menester contar con sistemas de modelado y simulación 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. En primer lugar, se dispone de todo un marco conceptual claramente estructurado para el modelado y simulación de sistemas en general, con múltiples aplicaciones y usos en áreas como la industrial, química, electromecánica, etc. En segundo lugar, 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 desarrollo, desde los más clásicos como el modelo estructurado en cascada, hasta modelos más orgánicos y menos rígidos como el RUP[1] o el que se tomará como referencia en este trabajo, y se explica en la sección 3.1, el Modelo de Craig Larman.

Surge a partir de las dos bases de conocimiento mencionadas anteriormente una interesante oportunidad 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 propone aquí tomar el método ampliamente estudiado y documentado de Craig Larman y diseñar un modelo de simulación que permita representar el funcionamiento de dicho método como proceso organizativo de desarrollo de software.

[1] Rational Unified Process.

1.3 Objetivos generales

1. Conocer el estado de la cuestión en torno al modelado y simulación de procesos de ingeniería del software.
2. Diseñar, construir y validar un modelo de simulación para la fase más procedimental del proceso de ingeniería de software representado en el Método de Larman: la fase de construcción.
3. Identificar posibles líneas de investigación a ser desarrolladas en el futuro con la intención de dar soporte a la toma de decisiones en los equipos de ingeniería del software, basados en la simulación.

1.4 Objetivos específicos

Con respecto al primer objetivo general "Conocer el estado de la cuestión en torno al modelado y simulación de procesos de ingeniería del software", se pueden plantear los siguientes objetivos específicos:

- Hacer una revisión de la literatura existente concerniente al modelado y simulación de procesos de ingeniería del software.
- Identificar los principales enfoques o corrientes de conocimiento que han tomado fuerza en los últimos años en el área de la simulación de procesos software.
- Identificar cuáles de esos enfoques de modelado y simulación son los más adecuados para acoplarse con las más recientes tendencias y estilos tanto organizativos como de desarrollo de software.

Con respecto al segundo objetivo general, "Diseñar y construir un modelo de simulación para la fase más procedimental del proceso de ingeniería de software representado en el Método de Larman: la fase de construcción", se plantean los siguientes objetivos específicos:

- Comprender el funcionamiento y características principales del método de desarrollo de software de Craig Larman (Larman, 2004).

- Identificar los elementos esenciales que describen el proceso de diseño y desarrollo de software en el Método de Larman.
- Describir la funcionalidad y elementos que intervienen en la fase de "construcción" del Método de Larman" para el desarrollo de software.
- Construir un modelo de simulación para la fase de "construcción" del Método de Larman, a partir de los elementos esenciales identificados.

Con respecto al tercer objetivo general "Identificar posibles líneas de investigación a ser desarrolladas en el futuro con la intención de dar soporte a la toma de decisiones en los equipos de ingeniería del software", se pueden plantear los siguientes objetivos específicos:

- Contrastar las posibilidades que abre el modelado y simulación de procesos software con las corrientes de desarrollo de software que han tomado fuerza en las últimas décadas.
- Identificar puntos de encuentro o de conexión posibles entre el modelado y simulación de procesos software y el desarrollo de software en el siglo XXI.
- Listar caminos de investigación sobre modelado y simulación de procesos software, viables al mediano y largo plazo.
- Proponer líneas específicas de investigación que sirvan para dar continuidad al presente trabajo.

1.5 Justificación

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 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 que permitan una mejor comprensión de los sistemas y su complejidad, con el menor consumo de recursos posible. 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, al ser ampliamente estudiado, documentado y tomado como referencia tanto en la teoría como en la práctica, representa un buen punto de inicio para comenzar un trabajo de investigación en torno al Modelado y Simulación de Procesos Software, por lo que es el escogido en este trabajo como modelo "ideal – de referencia" del proceso a ser simulado, con miras a ser una base para futuras comparaciones y contrastes con otros modelos del proceso software. Con respecto a la base conceptual escogida para el modelado y simulación de este sistema "real", se ha decidido trabajar con 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.

1.6 Metodología

La metodología a seguir durante este trabajo de investigación es híbrida. Un primer componente de esta metodología lo conforman todas las actividades necesarias para la conceptualización adecuada del contexto en el que se enmarca el proyecto: lectura de la bibliografía recomendada, análisis de áreas de investigación identificadas, consolidación de marco de conocimiento construido, y delimitación de la investigación entre el universo de posibilidades. Como segundo componente está el sub-marco metodológico para el diseño y construcción del modelo de simulación del proceso software, basado en los pasos descritos por (Sterman, 2000), para el proceso de modelado mediante Dinámica de Sistemas, que comprende:

- Revisión y comprensión de la información relacionada con el Método de Larman (Larman, 2004), sus pasos, fases, limitaciones, parámetros, etc.

- Realización de una extensa y completa descripción del sistema (sub-sistema del Método de Larman) a ser simulado.
- Desarrollo del modelo de simulación a partir de la exploración inicial del método y su funcionamiento, la identificación de las variables más relevantes y el estudio de la dinámica de estas en el transcurrir del tiempo.
- Realización de pruebas de verificación del modelo para comprobar la existencia de correspondencia entre lo que el modelo representa y el sistema real. Entre estas pruebas están: Existencia de las variables relevantes, correcta estructura del modelo, consistencia dimensional, comportamiento ante condiciones extremas, errores de construcción y modelado, etc.
- Definición de políticas y normas de uso y manejo del modelo para el apoyo a la toma de decisiones haciendo uso de la información que el modelo es capaz de arrojar.

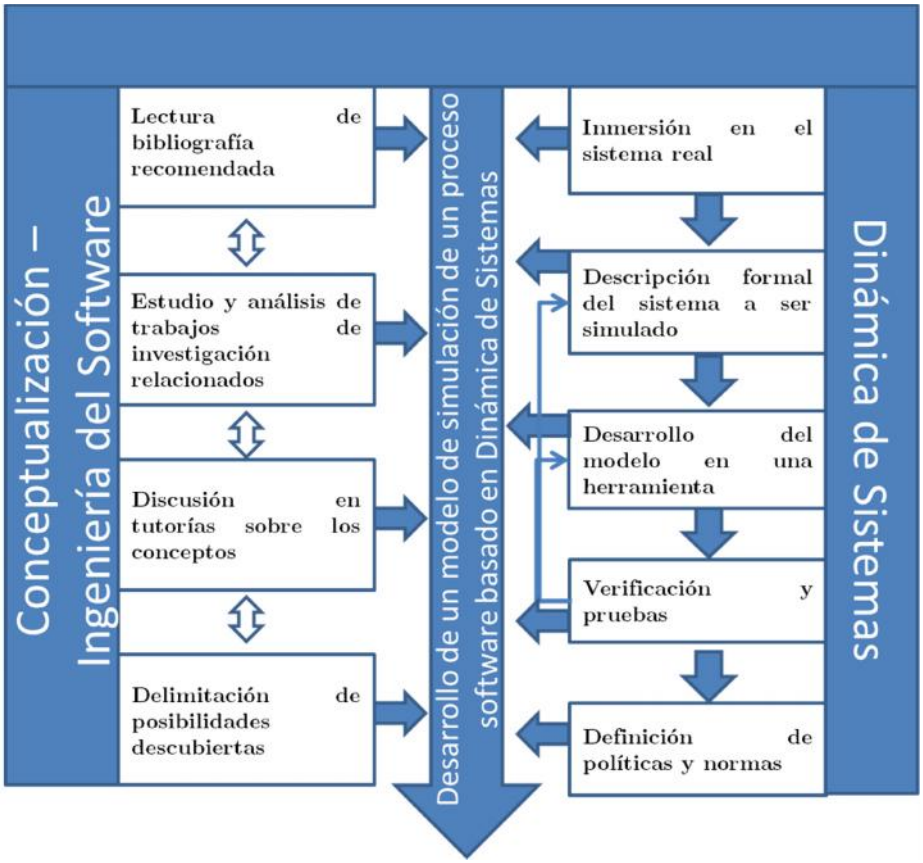

Figura 1 - Metodología de la Investigación

1.7 Alcance

El trabajo que se describe a continuación pretende llegar a presentar una documentada perspectiva de lo que es el modelado y la simulación de los procesos de ingeniería del software. Para ello, al finalizar el presente trabajo de fin de máster serán activos y productos de esta experiencia:

- La comprensión conceptual de los objetivos del modelado y la simulación de sistemas.
- La delimitación de enfoques de modelado y simulación que son útiles en el modelado y simulación de procesos software.
- La comprensión del Método de Larman, sus fases, componentes y características principales.
- La disposición de un modelo concreto basado en la Dinámica de Sistemas, con el que se puede experimentar sobre el proceso de desarrollo de software.
- La identificación de caminos de investigación en torno al modelado y simulación de procesos software.
- La identificación de proyectos de investigación que como continuidad del presente trabajo puedan abrir caminos de desarrollo de trabajo de investigación.
- El bosquejo de un marco conceptual inicial de modelado y simulación de procesos software que pueda ser desarrollado en profundidad en el futuro como productos del contraste de varias experimentaciones en el campo.

1.8 Estructura del trabajo de fin de máster

El presente trabajo se estructura en varios capítulos o secciones en los que se engloban bloques de contenido característicos para la comprensión total del tema objeto de investigación y el modelo construido.

La primera sección que aquí termina, ha sido introductoria al contexto en el que se enmarca este trabajo. Se presentó la motivación para la selección de este problema, los

objetivos generales y específicos planteados, la metodología a seguir y el alcance propuesto.

La segunda sección denominada "Estado de la Cuestión" presenta un resumen de la bibliografía más relevante que abarca el tema del modelado y la simulación de sistemas de desarrollo de software, haciendo énfasis en los tres enfoques usualmente utilizados en los últimos años para modelar estos procesos: la Dinámica de Sistemas, el enfoque de Agentes y Sistemas Multi-agente, y la Simulación de Eventos Discretos.

La tercera sección está a su vez compuesta por dos elementos: el primero es una breve explicación del Método de Larman y los principios conceptuales que le definen; el segundo elemento es el diseño y construcción del modelo de Dinámica de Sistemas para la simulación de un proceso de desarrollo de software. Como parte de este diseño del modelo se siguieron los pasos elementales de la Metodología de Diseño de Sterman, por lo que adicionalmente se presenta una descripción detallada del sistema en particular a ser simulado, sus correspondientes diagramas causales, y un estudio de escenarios que sirve para hacer una primera validación del modelo y su correspondencia con el sistema que pretende simular.

En la cuarta sección se hace uso del modelo construido para ejecutar distintas simulaciones, con diferentes configuraciones de parámetros y atributos, con la finalidad de identificar si los comportamientos expresados por el modelo se corresponden con las hipótesis que sobre efectos de variación de atributos se puedan plantear (Ejemplo: *Un mayor esfuerzo en la fase de análisis aminora el tiempo total de desarrollo del software*, etc.).

En la quinta y última sección se presentan las conclusiones a las que la experiencia de investigación durante el desarrollo del presente proyecto permiten llegar, así como las posibles vías de investigación que se abren, tanto de cara a continuar con la investigación iniciada como potenciales caminos en otras direcciones que se pudieron identificar.

2 Estado de la cuestión

En este capítulo se presenta inicialmente una breve introducción a lo que es el modelado y la simulación en general, seguido por una panorámica narrativa de los paradigmas de modelado y simulación que se pueden considerar para investigaciones y trabajos en el dominio del Proceso de Desarrollo de Software y algunos trabajos relacionados.

2.1 Modelado y Simulación

El modelado y la simulación de sistemas son dos conceptos generalmente acoplados, que implican una serie de actividades con importantes aportes para la toma de decisiones en torno a un sistema, realizar análisis situacionales, explorar comportamientos de los sistemas bajo ciertas condiciones y para la mejora de procesos en general (entre ellos procesos de ingeniería del software).

Por un lado, el "modelado" comprende la abstracción de un sistema en una representación de éste lo suficientemente amplia como para que sean observables las principales características del sistema real y en que se puede apreciar una versión simplificada de la realidad pero entendible en los mismos términos en que el sistema real era entendido. Es así que, por ejemplo, una maqueta arquitectónica es un modelo de una vivienda, un "fresco" de una pradera es un modelo de dicha pradera, o un bosquejo de un organigrama es un modelo de una estructura organizacional.

La "simulación", por su parte, implica la manipulación computacional de los parámetros de un modelo para conseguir representaciones del comportamiento de dicho modelo en un período de tiempo determinado y bajo ciertas condiciones iniciales y ambientales. Es así que se pueden hacer simulaciones de vuelo, simulaciones de crecimiento de poblaciones, o simulaciones del movimiento de los planetas.

Un modelo de simulación debe representar la dinámica de un sistema real y es una forma mucho más eficiente (en términos de consumo de recursos) de aproximar el

comportamiento del sistema, que la manipulación y experimentación del sistema real. En el caso que interesa en este trabajo, la manipulación de un sistema de desarrollo de software (personas, equipos, espacios, tiempos) implica costes y compromisos importantes que no suelen ser accesibles fácilmente, mientras que un modelo de simulación permitiría aproximar cómo se comporta este sistema sin necesidad de poner a trabajar el sistema real, ofreciendo importantes oportunidades de aprovechamiento en términos de economía, conocimiento y como una excelente herramienta para dar soporte a la toma de decisiones.

Tras haber realizado una revisión de la literatura reciente, relativa a la simulación de procesos software, es posible inferir cuáles son las áreas dominantes o que más frecuentemente han sido usadas como marco de comprensión y tratamiento de los problemas. Con una clara diferencia con respecto a los demás, el marco conceptual dominante para el modelado y la simulación de procesos es la Dinámica de Sistemas[2], que implica el uso de distintos programas de software que permitan implementar los *flujos* y *acumulaciones*, ambos, conceptos propios de dicho marco conceptual. También frecuentes, aunque en menor medida que la Dinámica de Sistemas, están el enfoque de "Agentes y Sistemas Multi-agente" y el enfoque de "Simulación de Eventos Discretos", de los que en (He Zhang, Kitchenham, & Jeffery, 2007) se presentan algunos ejemplos de uso.

Entre los restantes enfoques usados para modelado y simulación se cuentan los "enfoques híbridos" y otros que en algún aspecto guardan relación como la "Inteligencia Artificial" y el "Modelado de Procesos de Negocio (Business Process Modelling – BPM). De estos últimos destacan los enfoques híbridos, que según (Donzelli & Iazeolla, 2000) han sido menos frecuentes pero que presentan ventajas significativas en cuanto a la flexibilidad en el ajuste de parámetros de los modelos implementados, con lo que representan un significativo potencial de cara a las investigaciones futuras.

[2] La Dinámica de Sistemas resultó ser el paradigma para simulación con mayores resultados entre los años 1998 y 2007 (He Zhang et al., 2008).

La evolución histórica de la investigación en el tema del *Modelado y Simulación de Procesos de Ingeniería del Software* puede ser un indicador del comportamiento creciente que puede estarse dando en torno a la simulación de procesos; crecimiento reforzado por las infinitas capacidades tecnológicas de los últimos tiempos. Véase a continuación una breve descripción de los principales enfoques de modelado y simulación de procesos software.

2.2 Dinámica de Sistemas

La Dinámica de Sistemas, también conocida como SD por sus siglas en inglés (*Systems Dynamics*) es una metodología desarrollada en el seno del MIT (*Massachusetts Institute of Technology*) por el profesor Jay Forrester a mediados del siglo XX. Más que un método en sí, la Dinámica de Sistemas comprende una forma y estilo para abordar problemas de alta complejidad, es decir, en los que hay un número considerable de elementos interactuando y causando efectos entre sí. Es una metodología cuyo objetivo principal es estudiar la complejidad y comportamiento de estos sistemas complejos de realimentación en el transcurso del tiempo. (García, 2006) se refiere a la Dinámica de Sistemas como un enfoque para interpretar la realidad que, sin juzgar si es la forma correcta o la mejor, con certeza es una forma útil de abordar los problemas complejos que se planten en el siglo XXI, que no han sido suficientemente enfrentados por las formas tradicionales de solución.

La Dinámica de Sistemas es un paradigma de modelado y simulación en el que se construyen y manipulan "modelos de gestión", que en esencia, lo que pretenden es establecer que una alternativa X es mejor que otra alternativa Y, bajo ciertas condiciones. De esta manera, en el dominio del Proceso de Desarrollo de Software, ya se puede vislumbrar que la Dinámica de Sistemas se convierte en una herramienta para ver cómo los elementos intervinientes en el proceso (personas, tiempo, presupuestos, voluntades, disposiciones, etc.) afectan y definen que una configuración de estos parámetros sea mejor que otra, aportando importantes insumos para la toma de

decisiones en torno al proceso software, e incluso para la discusión sobre políticas y filosofías que deben regir el comportamiento organizativo de los equipos de desarrollo.

En la Dinámica de Sistemas se manejan varios conceptos que se deben entender muy bien para evitar desviar la atención y crear interpretaciones conceptuales erróneas. Los conceptos elementales son: "Niveles", "Flujos", "Fuentes y Sumideros", "Auxiliares", "Constantes" y "Conectores".

Los "Niveles" o "Variables Nivel", son acumulaciones que tienen la propiedad de describir el estado del sistema. Estas acumulaciones son importantes ya que al describir el estado del sistema, son la principal fuente de información para la toma de decisiones en torno al sistema simulado. A la vez tienen la propiedad de servir como "memorias" que almacenan el comportamiento del sistema en un período de tiempo dado. En el dominio de la ingeniería del software, los niveles se pueden usar para representar la acumulación en el transcurso del tiempo de entidades como artefactos-software, defectos, líneas de código o esfuerzo.

Los "Flujos" o "Variables-tasa" siempre se usan con los niveles, de los que son la causa de variación; por tanto, en un sistema pueden haber tanto *flujos de entrada* como *flujos de salida* de un nivel (Véase la Figura 2), y son éstos los que determinan la forma como el nivel varía (se incrementa o decrementa) en el transcurso del tiempo. Una forma de entender los flujos es pensando en ellos como corrientes de agregación o disminución de un nivel durante un período de tiempo determinado. En el dominio de los sistemas en general, ejemplos típicos de flujos serían: la cantidad de agua que se vierte en un contenedor por segundo, la cantidad de personas que pasan una frontera incrementando una población de inmigrantes en un período de un mes, o más cercano al campo de la ingeniería del software, un flujo puede ser el desarrollo de artefactos, generación de defectos o errores, la distribución del personal y la cantidad de tareas de codificación que genera un analista de software durante su proceso de análisis (Ejemplo: 5 tareas por día, etc.). En la Figura 2 se presenta de forma gráfica cómo se relacionan

los flujos, niveles, variables y fuentes en la Dinámica de Sistemas, y en la Figura 7 se presenta un ejemplo completo e instanciado con niveles, flujos y variables.

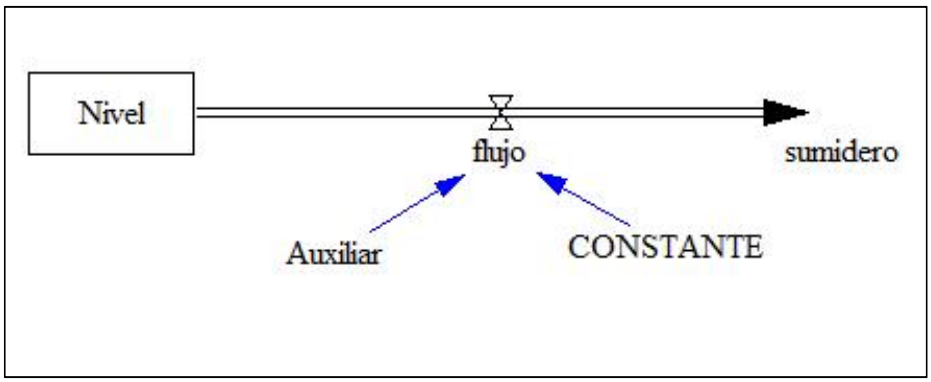

Figura 2 - Niveles, flujos, constantes y auxiliares en la Dinámica de Sistemas

Las "Fuentes" y "Sumideros" representan los límites del sistema. En el dominio de la ingeniería del software un ejemplo de una fuente puede ser una agencia de contratación de personal que nutre la plantilla laboral. Igualmente, un ejemplo de un sumidero puede ser la cantidad de artefactos que le son entregados al cliente.

Las "Auxiliares" son variables que se usan para habilitar cálculos que han de hacerse en el sistema de simulación. En el caso de la ingeniería del software, un ejemplo de variable auxiliar son elementos que se usan para representar las demoras que se pueden presentar en la entrega de artefactos a los clientes (demoras aleatorias, que siguen un patrón específico, estocásticas, de tipo *lookup*, etc.).

Las constantes que se incorporan al sistema son usadas para representar factores que son determinantes e invariables en el modelado del sistema. Se usan para calibrar el modelo con la intención de que se parezca a un contexto o escenario específico de interés. Un ejemplo de una constante en la ingeniería del software es la cantidad inicial de personas de un equipo de desarrollo.

Los conectores que se usan en Dinámica de Sistemas son útiles en la representación de flujos de información entre los elementos que existan en el sistema. De esta manera, cuando una variable está conectada con otra, la conexión (representada por una flecha) indica que la de origen tiene algún tipo de influencia sobre la de destino. En el dominio

de la ingeniería del software, una conexión podría ser la relación existente entre la tasa de entrega de un producto (que es una constante) y el flujo de entrega del producto en sí mismo. (Véase la Figura 3).

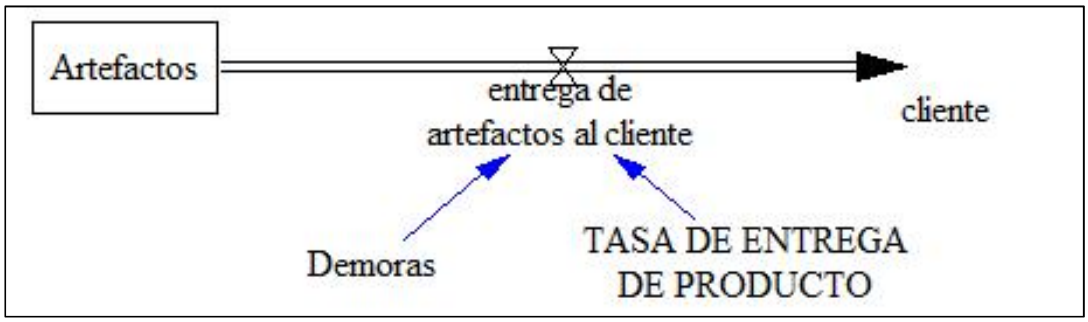

Figura 3 - Ejemplo de interacción entre niveles y flujos en la Dinámica de Sistemas

Los "Ciclos de realimentación" son elementos básicos de la estructura de los modelos en Dinámica de Sistemas, y corresponden a las relaciones de causa y efecto entre un grupo de elementos en los que se observa que una alteración o modificación en una parte del sistema puede producir cambios en otra parte; cosa que puede ocurrir de manera recíproca, aunque no necesariamente por el mismo camino. Los ciclos de realimentación pueden ser positivos, cuando el efecto de la variación de un elemento se propaga por todo el ciclo de realimentación reforzando esta variación; o pueden ser negativos, que es cuando el efecto de la variación en un elemento se propaga causando variaciones en sentido contrario en el mismo elemento. (Sterman, 2000).

El uso de todos los elementos nombrados anteriormente, como miembros de un sistema en el que interactúan, permite hacer una representación bastante buena y completa de un sistema real a través de un modelo, del que se puede entonces inferir información extrapolable al funcionamiento del sistema real, con lo que se obtiene una ganancia significativa a un bajo coste si se compara con el coste que implicaría obtener esta información a partir del sistema real. Algunos de los trabajos de simulación desarrollados que guardan relación con el enfoque de la ingeniería del software, se pueden encontrar en (Raffo, 2012), (Vicente, 2012a),(Khosrovian, Pfahl, & Garousi, 2008), (De Sousa Coelho, Braga, & Ambrósio, 2013), (Ferreira, Collofello, Shunk, &

Mackulak, 2009), (H. Zhang, Jeffery, Houston, Huang, & Zhu, 2011), (Cao, 2010) y (Wu & Yan, 2009); de los que hablaremos más adelante en este capítulo.

En (Vicente, 2012b), se presenta un caso en el que se usó Dinámica de Sistemas para simular la especificación de requisitos de un sistema de información para una granja dedicada a la compra de animales, su manutención y venta. En este trabajo se sigue una metodología compuesta por los siguientes pasos:

1) Formulación del problema;
2) Modelamiento cualitativo de la estructura básica del sistema;
3) Modelamiento cuantitativo;
4) Modelamiento enriquecido del sistema; y finalmente
5) Transformación de modelos cualitativos y cuantitativos.

Como se puede intuir al analizar la metodología seguida, este trabajo consistió en la construcción de un modelo de simulación puntual, fundamentado en la Dinámica de Sistemas, para un estudio de caso específico, en el que como producto de todo el proceso de modelado, se alcanzó una extensa especificación semántica de los requisitos para el sistema de información de la granja. En este caso el uso de la Dinámica de Sistemas fue un complemento al proceso de Ingeniería de Requisitos; su carácter interdisciplinario e iterativo permitió abordar el conocimiento y las expectativas de los *stakeholders*, y representarles en diagramas, a partir de los que surgieron elementos aprovechables para la especificación de requisitos, como es el caso de los pares sustantivo-verbo, nivel-flujo, variable-derivada, atributo-método; y modelos matemáticos que permitieron identificar nuevos elementos de interés. Este caso es un ejemplo de cómo a partir de una representación abstracta del sistema, haciendo uso del lenguaje de la Dinámica de Sistemas, puede surgir información nueva que no era evidente con la simple observación del sistema real, lo que es aprovechable tanto para lograr una mejor y más completa comprensión del sistema, como para dar soporte y proveer de insumos al debate sobre decisiones que deban tomar en torno al sistema

real; aspecto fundamental que ha motivado y se ha convertido en un objetivo de la investigación que se presentará en este trabajo.

En (Khosrovian et al., 2008) se presenta un modelo de simulación de procesos software con una estructura personalizable para procesos específicos de las organizaciones, por medio del uso de estructuras genéricas o macro-patrones. La principal característica de interés de este modelo es que constituye un modelo genérico que permite calibrar los una serie de parámetros de modo que al usarlo para representar un proceso software de una organización, éste se parezca más a los datos con los que la organización pueda contar, y por ende a la realidad representada. Este trabajo resulta de particular interés para esta investigación por hacer uso del software Vensim, con lo que prueba su manejabilidad y usabilidad en el dominio del modelado de procesos software; además se trata de un estudio de simulación de un proceso software estructurado y conocido: el V-model; significando un precedente importante para este trabajo en cuanto a que es un modelo conciso y específico de un proceso software, tal como el que se presenta en este trabajo.

El trabajo de (Khosrovian et al., 2008) toma en cuenta parámetros para el modelo de simulación correspondientes a tres elementos fundamentales del proceso: el proceso en sí mismo, las personas que intervienen en él y el producto desarrollado. Los autores logran con esto una significativa abstracción del proceso que, al representar tal gama de aspectos que tienen alta influencia sobre el proceso, se aproxima muy bien a la realidad y se convierte entonces en una buena fuente de información sobre el proceso de desarrollo de software como un todo. A continuación, en la Figura 4, se presentan los parámetros usados en este modelo, y adicionalmente el sub-modelo en que se usó: C-P (proceso), C-Q (Calidad del producto) y C-W (Distribución del esfuerzo), resultando de especial interés si se toma en cuenta que en el modelo que se va a presentar también se toman en cuenta una serie de parámetros de distinto tipo, véase la Tabla 8.

	Parameter Name	Attribute	Type	View
1	Verify code or not	Process	Input	C-P
2	# of modules per subsystem	Product	Input	C-P
3	Code doc quality threshold per size unit	Project	Input	C-Q
4	Required skill level for code dev	Project	Input	C-W
6	Developers' skill level for code dev	People	Input	C-W
8	Maximum code ver. effectiveness	Process	Calibrated	C-P
9	Maximum code ver. rate per person per day	Process	Calibrated	C-P
12	Minimum code fault injection rate per size unit	Product	Calibrated	C-Q
14	Code rework effort for code faults detected in CI	Product	Calibrated	C-Q
16	Code rework effort for code faults detected in IT	Product	Calibrated	C-Q
18	Initial code dev. rate per person per day	People	Calibrated	C-W
19	Initial code ver. rate per person per day	People	Calibrated	C-W
20	Code doc size (actual)	Product	Output	C-P
22	Code development rate (actual)	Process	Output	C-P
24	Code faults undetected	Product	Output	C-Q
26	Code faults corrected	Product	Output	C-Q
28	Code ver. effort (actual)	Process	Output	C-W

Figura 4 - Parámetros usados en el modelo de simulación de (Khosrovian et al., 2008).

En el trabajo de (De Sousa Coelho et al., 2013), se desarrolla un modelo mediante Dinámica de Sistemas para simular el proceso de inspección de software (y su documentación). La motivación de este trabajo está basada en que la detección temprana de defectos en el proceso de desarrollo de software representa múltiples beneficios tanto para el equipo de desarrollo como para el cliente en sí mismo, ya que la detección en las últimas fases de desarrollo conlleva un trabajo mucho mayor para reparar los defectos, cosa que implicaría re-analizar, re-diseñar, re-codificar, y hacer pruebas nuevamente para probar la calidad del software. A su vez, los errores detectados en fases tardías suelen tener una mayor capacidad para reproducirse y generar nuevos errores o empeorar los ya existentes.

Como se puede observar en la Figura 5, el modelo consiste de tres niveles o acumulaciones que describen el estado del sistema en un momento dado: la cantidad de páginas inspeccionadas, los defectos no encontrados y el número total de defectos detectados.

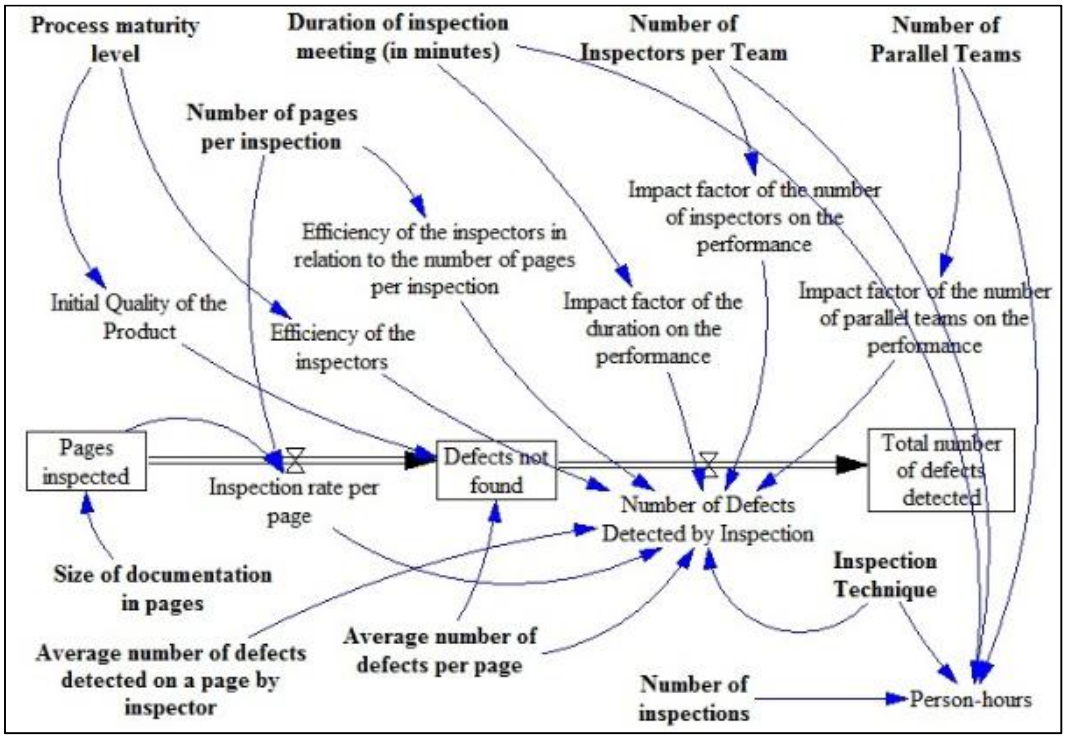

Figura 5 - Un ejemplo de modelo de simulación creado según la Dinámica de Sistemas

Al observar el modelo se pueden apreciar las relaciones existentes en el sistema que dan vida al proceso de inspección, y los parámetros que permiten implementar el modelo para este caso: *Nivel de madurez del proceso*, *Número de equipos paralelos*, *Número de inspectores por equipo*, *Promedio de Defectos por Página*, *Promedio de Defectos que un Inspector encuentra en una Página*, y hasta parámetros más cuestionables como la eficiencia de los inspectores o la calidad del producto. (De Sousa Coelho et al., 2013) argumentan que su modelo es una herramienta para dar soporte a la toma de decisiones, ya que permite predecir los resultados del proceso de inspección a partir de unos parámetros específicos que los *stakeholders* pueden manipular para estimar escenarios que no sólo ayudan a conocer posibles comportamientos del sistema sino una mejor comprensión del sistema en sí. Aunque a un nivel mucho más específico en una fase del proceso software, este trabajo nos da una idea de algunos parámetros que tienen influencia en el proceso, alimentando la visión general del proceso que se presentará y simulará la sección 3.5.

En (Ferreira et al., 2009) se presenta un modelo basado en Dinámica de Sistemas desarrollado para ayudar a los gestores de proyectos a comprender el complejo impacto que tiene la volatilidad de los requisitos en un proyecto de desarrollo de software. El foco del trabajo aquí presentado está en demostrar por medio del modelo de simulación la importancia en cuanto a costes, planificación y calidad que tiene la volatilidad de los requisitos. La volatilidad es entendida en este trabajo como el crecimiento de los cambios en los requisitos durante el ciclo de vida del proceso software, y los autores sostienen que un valor importante de este modelo de simulación es que permite a los interesados entender el proceso de ingeniería de requisitos y el impacto de la volatilidad de los requisitos.

El trabajo de (Ferreira et al., 2009) permite estimar en cierto sentido cómo afecta la incertidumbre (y por ende volatilidad) en el proceso de desarrollo de software, además de que permite entender las relaciones entre los factores que contribuyen con la volatilidad. Por medio de la configuración de distintos parámetros, el modelo permite además estimar el impacto esperado de distintas combinaciones de factores en el proceso completo de desarrollo de software (incluyendo la ingeniería de requisitos), constituyendo distintos escenarios posibles que se pueden presentar en el sistema. Finalmente, este trabajo enriquece su presentación incorporando variables estocásticas estimadas a partir de exhaustivas encuestas que permiten aportar probabilísticamente un poco más de realidad al modelo representado.

Este último trabajo de (Ferreira et al., 2009) es una orientación relacionada con los aspectos que se evaluarán en la Sección 4 sobre el modelo presentado en este trabajo, en el que desde un punto de vista crítico validaremos la representatividad del modelo sobre el sistema real y trataremos de ver como se ve afectado el modelo cuando ciertos parámetros son modificados, permitiendo ver en cierto modo la volatilidad de éste con respecto al sistema real.

A partir del análisis hecho en esta subsección sobre los trabajos que usan la Dinámica de Sistemas para modelar el proceso software, o parte de él, se puede observar que

aunque todos representan aportes significativos, están en cierta forma enfocados en uno de dos aspectos: o se enfocan en ver como es el proceso de gestión relativo al proceso software, con todas las complejidades y parámetros que involucra; o se enfocan en una representación más exacta del proceso estructurado, el *paso a paso*, que constituye los modelos formales (aunque menos reales) de procesos software. El trabajo de investigación que se presenta acá busca ser un punto inicial que incorpora ambos enfoques, valiéndose de un software de modelamiento de sistemas complejos para representar, modelar y simular el proceso software como un todo, con aspectos estructurados (se rigen por un proceso formal) pero complejos (en los que intervienen múltiples parámetros de distinto tipo).

2.3 Agentes y Sistemas Multi-agente

El enfoque de agentes se ha consolidado como una herramienta muy útil para modelar sistemas complejos, y se centra, como es de esperar, en las entidades computacionales denominadas "agentes". Dar una definición precisa del término "agente" no es tarea fácil, sin embargo, en la literatura se pueden encontrar distintas definiciones de este término; por ejemplo, en investigaciones de los campos sociales y de la biología, un agente se define como un objeto con un "mecanismo indirecto"(Ramon & Villegas, 2001); para los físicos, un agente es una entidad capaz de influenciar otros objetos que están a su alrededor (Izquierdo, 2008). En el caso del Modelado y Simulación de Procesos Software, es esta segunda definición la que más sentido tiene tomar como referencia ya que bajo esta perspectiva un agente se puede entender como una entidad-objeto con reglas de comportamiento para la interacción con otros agentes y con el entorno en el que se encuentra presente. En cualquier caso, mediante la simulación basada en agentes, el modelador reconoce de manera explícita que los sistemas complejos (y particularmente los que involucran personas), son producto de comportamientos individuales y de sus interacciones (Izquierdo, 2008).

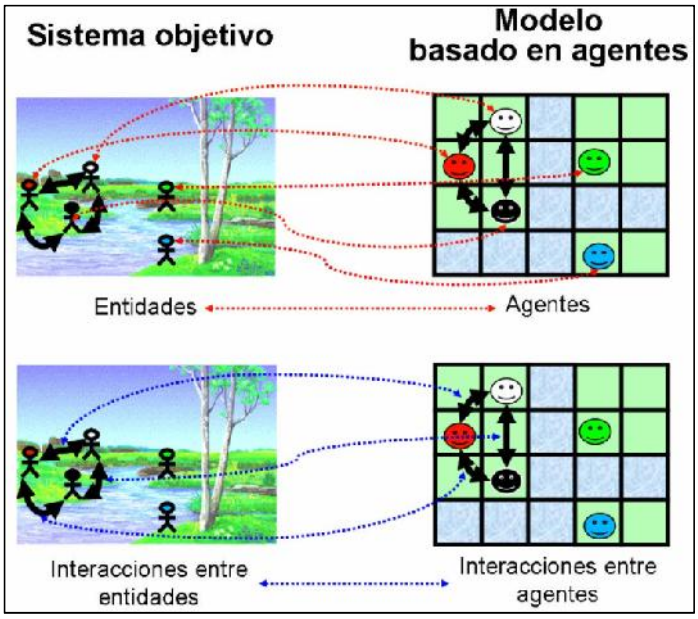

Figura 6 - Un ejemplo de modelo de simulación creado bajo el enfoque de Agentes y Sistemas Multi-agente

Un sistema multi-agente es, entonces, un conjunto de agentes con reglas establecidas de comportamiento, que interactúan en un entorno determinado y bajo ciertas condiciones ambientales e individuales. La importancia crucial de un modelo de simulación de un sistema basado en agentes es que debe estar basado en la *correspondencia real* entre el sistema en sí y el modelo. De esta manera, las fronteras que definen los elementos básicos del sistema real, deben corresponderse con las fronteras que definen los agentes del modelo, y las interacciones que se dan entre los componentes del sistema real deben corresponderse con las interacciones que se dan entre los agentes del modelo. Como argumenta (Izquierdo, 2008), la correspondencia entre el modelo de agentes y el sistema real "*es capaz de aumentar el realismo y el rigor científico de los modelos formales*".

Las principales características de los sistemas que comprenden varios agentes son:

- Los agentes del sistema son autónomos, heterogéneos e independientes.
- Cada agente sigue sus propias metas y objetivos.
- Pueden ser capaces de interactuar con otros agentes y con su entorno.

- Tanto los agentes como el entorno pueden evolucionar con el transcurso del tiempo de la simulación.
- Sus agentes tienen alta adaptabilidad.
- Sus objetivos no son fijos, sino que pueden cambiar en función de la interacción entre agentes y con el entorno.

En cuanto a los sistemas que son viables candidatos a ser modelados bajo el enfoque de agentes, (Izquierdo, 2008) especifica que este enfoque es especialmente relevante para:

- Sistemas con componentes heterogéneos cuya heterogeneidad no ha sido estudiada en profundidad.
- Sistemas en los que los componentes individuales son capaces de aprender individual y colectivamente y así adaptarse al entorno.
- Sistemas en los que el espacio geográfico parezca tener relevancia e importancia en el comportamiento del sistema.
- Sistemas en los que existan redes sociales de interacción o en los que sea evidente el posible surgimiento de éstas.
- Sistemas en los que se desee analizar en profundidad las relaciones que puedan existir entre los atributos/comportamientos de los individuos y las propiedades globales del grupo de individuos o sociedades.

Entendido esto, un modelo de un sistema de desarrollo de software (basado en agentes), es un modelo en que se identifican agentes que intervienen en el proceso de desarrollo de software, y reglas de comportamiento e interacción para estos agentes. Ejemplos de trabajos en este campo son los presentados en (Spasic & Onggo, 2012), (Saoud, 2002), (Cherif & Davidsson, 2009) y (Agarwal & Umphress, 2010).

En el trabajo de (Spasic & Onggo, 2012) se presenta el desarrollo, calibración y validación de un modelo de simulación basado en ***agentes*** representando un proceso de desarrollo de software. Por un lado aportan una función del esfuerzo práctico para estimar el comportamiento de los desarrolladores, y por otro lado presentan el proceso

de construcción del modelo en sí, haciendo uso de los datos ya disponibles para los departamentos de desarrollo. Específicamente, se enfocan en la fase de "construcción" de un proceso RUP usado en un departamento de desarrollo de software geográficamente distribuido en la organización AVL.

El modelo de simulación que se presenta en la sección 3 del presente trabajo es un claro complemento al trabajo de (Spasic & Onggo, 2012) ya que también es un modelo de simulación enfocado en la fase de construcción de un proceso software. Además, tienen una gran similitud contextual ya que en el caso de (Spasic & Onggo, 2012) se basaron en un proceso RUP, mientras que en el caso del presente trabajo, se basa en el proceso descrito en el Método de Larman, que es una derivación del RUP. Quizás la principal diferencia se evidencia en el hecho de que nuestro modelo pretende servir a la comprensión del sistema simulado desde un punto de vista en que de soporte a la toma de decisiones organizativas, por lo que las pruebas y conclusiones se enfocarán a discutir una perspectiva sistémica de las observaciones del modelo de simulación y de sus resultados.

El enfoque de agentes parece ser de bastante utilidad, y, a primera vista parece tener un potencial de importancia que no puede ser ignorado. A efectos de este trabajo, el modelo que se construirá está fundamentado en la Dinámica de Sistemas, sin embargo ha sido interesante presentar el enfoque de agentes con la intención de proponer la creación, con posterioridad a este trabajo, de un modelo basado en agentes, tomando en cuenta las mismas condiciones y parámetros que el modelo aquí presentado ha tomado en cuenta. A partir de este modelo alternativo y su comparación con el de Dinámica de Sistemas, pudiera surgir un debate rico en criterios y contrastes para alimentar la investigación contextual sobre *marcos metodológicos para modelado y simulación de procesos software* que enmarca este trabajo.

2.4 Simulación de Eventos Discretos

Existen varios trabajos que han usado el enfoque de los eventos discretos para simular procesos de desarrollo de software. Como argumenta (Martin & Raffo, 2001), el enfoque

de los sistemas discretos es muy útil para simular actividades de procesos de ingeniería del software. Ejemplos de trabajos en este campo son los presentados en (Ioana Rus, Collofello, & Lakey, 1999), (SeungHun Park, Choi, Yoon, & Bae, 2008), (I. Rus, Neu, & Munch, 2003) y (H. Zhang et al., 2011).

En el enfoque de eventos discretos, el tiempo avanza de evento a evento, y permite la representación de tareas específicas de procesos. Los modelos bajo este paradigma facilitan la descripción de los "artefactos únicos de un proceso" por medio de los atributos descriptivos específicos. Sin embargo, estos modelos al igual que los basados en "estados" del sistema, tienen el problema de que el tamaño de paso de la simulación no es constante, sino que algunos eventos pueden llevar más tiempo que otros, por lo que una macro descripción del sistema o proceso puede resultar bastante imprecisa, aunque micro descripciones de eventos resulten bien representados.

(Martin & Raffo, 2001) presentan un modelo híbrido en el que se complementan un elemento bajo el enfoque de la Dinámica de Sistemas y un elemento de simulación basado en el enfoque de los eventos discretos. El enfoque discreto se usa en este caso para representar los detalles de los procesos, como: flujo de actividades, tiempos de espera, simultaneidad, etc.

(Ioana Rus et al., 1999) presentan un trabajo similar al de (Martin & Raffo, 2001) en el sentido de que también se trata de un modelo híbrido compuesto en parte bajo el enfoque de la Dinámica de Sistemas y en parte bajo el enfoque de la Simulación de Eventos Discretos. Este trabajo tenía la intención principal de presentar un simulador que sirviera de soporte a la toma de decisiones; sentido en el que los autores comentan que su parte dinámica tiene una variedad de usos tan diversa que sirve tanto para el entrenamiento de gestores de proyectos software como para la planificación de proyectos; mientras que el componente discreto, al ser más detallado, es de gran utilidad en el control y monitorización. En este caso específico se presenta en detalle la implementación de un modelo de eventos discretos del proceso software, para prestar apoyo a un proyecto específico: el proyecto del ejercito *Crusader*. El enfoque que se

siguió permitió modelar un *Computer Software Configuration Item* (CSCI), mejor entendido como un subsistema para el aprendizaje sobre la dinámica de los defectos, la fiabilidad y otros parámetros del proceso. El objetivo principal del modelo es predecir, hacer seguimiento y controlar los defectos y fallos sobre todo el desarrollo del proyecto *Crusader* (aunque sólo se implementó para la fase de Diseño Preliminar", permitiendo además monitorear y controlar los costes de proyectos y cumplimiento de planes en el transcurrir del tiempo.

En (I. Rus et al., 2003) se desarrolló y presentó en el año 2003 una Metodología Sistemática para el Desarrollo de Modelos de Simulación de Eventos Discretos de los Procesos de Desarrollo de Software. El esfuerzo que resalta de esta propuesta metodológica está en que trata de proveer a la comunidad científica de un marco de diseño de simuladores de procesos software no pensado para diseños específicos sino más generales y menos *ad hoc*. La metodología propuesta considera el desarrollo de un modelo de simulación nuevo y sin componentes que reutilizar, toma en cuenta que el ciclo de vida de un modelo de simulación es similar al del software, principalmente consistiendo de tres fases: *desarrollo*, *despliegue* y *operación* (que incluye *mantenimiento* y *evolución*); y considera que las actividades a desarrollar en todas estas fases pueden tener distintos tiempos, orden y dependencias, por lo que el ciclo de vida puede tener distintas formas, como pueden ser: cascada, de iteraciones o incluso de proceso ágil. Este modelo considera dos categorías principales de actividades en el proceso software: las actividades de ingeniería y las actividades de gestión.

Entre las *Actividades de Ingeniería* están:

- Identificación de requisitos y especificación para el modelo a construir.
- Análisis y especificación del proceso modelado.
- Diseño del modelo.
- Implementación del modelo.
- Verificación y validación durante el desarrollo.

Entre las *Actividades de Gestión* están:

- Planificación y monitorización del desarrollo del modelo.
- Medición sobre el modelo y del proceso de desarrollo del modelo.
- Gestión del riesgo.

En el proceso de desarrollo de un modelo de simulación de un proceso software según la metodología de (I. Rus et al., 2003) se ven involucrados varios roles durante todo el proceso. En la fase de desarrollo los más involucrados son el desarrollador y el cliente, mientras que en la fase de despliegue hay una intervención mayor por parte del equipo de gestión y supervisión.

La principal diferencia entre la metodología de (I. Rus et al., 2003) y la de (Sterman, 2000), está en el hecho de que mientras la de (I. Rus et al., 2003) es analítica y detallada, la usada en este trabajo de investigación (Sterman, 2000) es una metodología inspirada y creada a partir del concepto y filosofía de los *sistemas*, por lo que considera desde su fase inicial el conjunto de elementos como un todo y trata de manejarlos y concebirlos durante todo el proceso como un "*todo que es más que la suma de sus partes*". La metodología de (Sterman, 2000), que ha inspirado la investigación realizada, parte una conceptualización general, pero no simplista del sistema real, y comienza a "*subir*" durante el proceso de abstracción manteniendo las características del sistema que le describen y le hacen único, sacrificando sólo aquellas piezas de información que no son esenciales. Finalmente, el trabajo deriva en un modelo que trata de ser "sistémico" con un grado suficiente de abstracción que le permita ser representativo, en el que se incorporan todos los elementos esenciales, no en términos de agentes independientes e interactuantes (que sería el caso de (I. Rus et al., 2003)), sino en términos de elementos interrelacionados que están en constante relación y dinamismo y entre los que *fluye* información que al pasar de un lado a otro modifica el estado general del sistema, Una última diferencia es la forma como se aprovecha la información que un modelo de Dinámica de Sistemas maneja, ya que en este caso la interpretación de resultados se hace en base a las modificaciones y efectos que unos elementos tienen sobre otros en el transcurso del tiempo de la simulación, permitiendo sugerir no sólo hipótesis sobre el sistema sino descubrir comportamientos y relaciones

que no son evidentes si se observa sólo el sistema real, cuya complejidad suele hacer de muchos aspectos algo inobservable.

En otro trabajo, (SeungHun Park et al., 2008) presentan un enfoque para la construcción de modelos de simulación de procesos software a partir de un meta-modelo y las reglamentarias reglas de transformación. En este trabajo, se parte de un modelo de proceso software basado en un Meta-modelo de Ingeniería de Procesos Software (SPEM[3]), para derivar un modelo de simulación híbrido basado en una Especificación del Sistema de Eventos Discretos (DEVS-hybrid[4]). En otras palabras, la propuesta es derivar un modelo de simulación del proceso software a partir de un modelo descriptivo del proceso. Por una parte, SPEM es un meta-modelo para la definición de procesos y sus componentes; por la otra, se puede entender como un estándar para el modelado de procesos que hace uso de UML como lenguaje de modelado. El modelo de simulación DEVS-Hybrid está basado en el formalismo DEVS-Hybrid (SeungHun Park et al., 2008).

El proceso de derivación (Ver Figura 7) del modelo de simulación propuesto por (SeungHun Park et al., 2008) consiste de tres pasos fundamentales: Mapeado (Figura 8), Modelado y Transformación. En el paso de Mapeado se "mapean" los elementos del SPEM y el DEVS; seguidamente, en el paso de Modelado, se desarrolla un modelo descriptivo de procesos haciendo uso de UML; y finalmente, en el paso de Transformación, se transforma el modelo descriptivo del proceso en un modelo de simulación DEVS-Hybrid, por medio de la aplicación de las reglas de transformación.

[3] Software Process Engineering Meta-model (SPEM)-based software process model.
[4] Discrete Event System Specification (DEVS)-Hybrid simulation model.

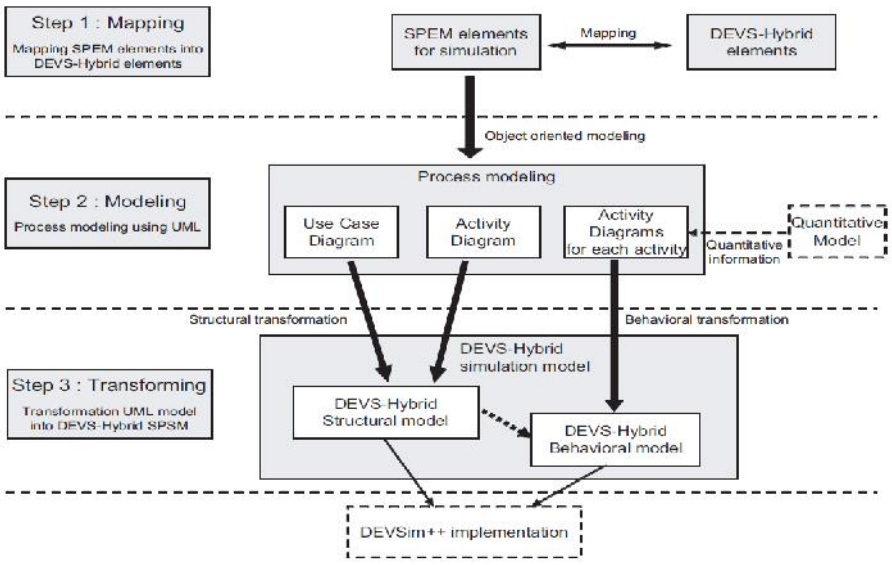

Figura 7 - Proceso de derivación de un modelo de simulación híbrido (SeungHun Park et al., 2008).

El principal beneficio argumentado por los autores en (SeungHun Park et al., 2008) es que el coste de diseño de un modelo de simulación del proceso software se ve considerablemente reducido por permitir la transformación automática de un modelo descriptivo en un modelo de simulación por medio de la aplicación de las reglas de transformación.

Simulation Elements	**SPEM**	**DEVS-Hybrid**
Structural Elements	WorkDefinition, Phase	Coupled model
	Activity	Atomic model
	WorkProduct, Document	Input value(X), Output value(Y), Phase output value(Y^{phase})
Behavioral Elements	Step	State(S)
	Transition	External transition function (δ_{ext},) Internal transition function (δ_{int})
		Output function (λ)
		Time advance function(ta)
		Phase event condition function(C_{phase})

Figura 8 - Mapeado de Elementos SPEM y DEVS-Hybrid

2.5 Otros enfoques.

Ejemplos de trabajos bajo otros enfoques son los presentados en (Donzelli & Iazeolla, 2000), (Martin & Raffo, 2001), (Filho & Rocha, 2010), (Rodríguez, Ruiz, Riquelme, & Harrison, 2011), (Raffo, Vandeville, & Martin, 1999) y (Seunghun Park & Bae, 2011).

(Raffo et al., 1999) presenta un trabajo claramente enfocado en el diseño y uso de modelos de simulación de procesos software como una herramienta para alcanzar un alto nivel de capacidad de proceso de desarrollo de software. Esta mejora es observable por medio del posicionamiento de la compañía de desarrollo de software bajo estudio, en los niveles 4 y 5 del marco de referencia *Capability Maturity Model* (CMM). Estas herramientas de simulación fueron desarrolladas por el *Northrop Grumman´s SBMS*[5] y la Universidad del Estado de Portland PSU[6], con el apoyo y patrocinio del SERC[7], y consistían de ***modelos de simulación estocásticos*** del proceso de desarrollo de software. Estos modelos de simulación resultaron de gran utilidad en el sentido de que eran capaces de proporcionar una cantidad grande de información cuantitativa que a su vez permitía hacer análisis cuantitativos del sistema y de medidas descriptivas del sistema, como son: "Coste de desarrollo", "Calidad del Producto" y "Planeación del Proyecto". La bondad de la manipulación de estos modelos y de esta información cuantitativa viene en que son de gran ayuda en la evaluación del riesgo o incertidumbre asociada con las alternativas de cambio del proceso. El enfoque de simulación aquí presentado sirvió de apoyo para la implementación con éxito de los niveles 2 y 3 del modelo de mejora de procesos CMM, y particularmente los aspectos estocásticos y analíticos de este enfoque sirven de apoyo a las prácticas clave relacionadas con la Gestión de Procesos Cuantitativos y la Gestión de Calidad del Software del nivel 4 del modelo CMM. Adicionalmente, y como reafirmación den la utilidad de la simulación en la mejora de procesos software, este trabajo contribuyó en el sentamiento de las bases

[5] SBMS: Surveillance and Battle Management Systems (SBMS Melbourne, Florida) .
[6] PSU: Portland State University.
[7] SERC: Software Engineering Research Center.

para la Gestión del Cambio de Procesos y Tecnologías, relativas al nivel 5 del CMM, y su mejora continua.

2.6 Síntesis de trabajos revisados

A continuación, en la Tabla 1, se resumen los trabajos revisados como parte de la conceptualización necesaria para llevar a cabo este trabajo y como parte de la exploración del estado de la cuestión en torno al modelado y la simulación de procesos software. Para facilitar la representación de estos trabajos en una tabla, se presentará la referencia, que se puede consultar en la bibliografía, se indicará el paradigma con el que se relaciona el trabajo, se indicará el año de publicación, por considerarse importante para comprender la vigencia que tiene, y finalmente se señala si el aporte a este trabajo es en forma de un trabajo relacionado o si es de tipo conceptual.

Para la sección de la Tabla 1, en la que se señala el paradigma de modelado relacionado se usan los siguientes acrónimos:

- D.S. → Dinámica de Sistemas.
- A.S.M → Agentes y sistemas multi-agente.
- S.E.D. → Simulación de Eventos Discretos.
- O.E. → Otros enfoques.

Tabla 1 - Resumen de trabajos referenciados en el estado de la cuestión.

Referencia	**Paradigma relacionado**				**Año**	**Aporte**	
	D.S.	A.S.M.	S.E.D.	O.E.		Solución a nivel teórico	Solución a nivel teórico/práctico
(He Zhang et al., 2007)		X	X		2007		X
(Donzelli & Iazeolla, 2000)				X	2000	X	X
(He Zhang, Kitchenham, & Pfahl, 2008)	X				2008	X	X
(García, 2006)	X				2006	X	
(Sterman, 2000)	X				2000	X	
(Raffo, 2012)	X				2012		X
(Vicente, 2012a)	X				2012	X	X
(Khosrovian et al., 2008)	X				2008		X
(De Sousa Coelho et al., 2013)	X				2013		X
(Ferreira et al., 2009)	X				2009		X
(He Zhang et al., 2014)	X				2014		X
(Cao, 2010)	X				2010		X
(Wu & Yan, 2009)	X				2009		X
(Ramon & Villegas, 2001)		X			2001	X	X
(Izquierdo, 2008)	X	X			2008	X	X
(Spasic & Onggo, 2012)		X			2012	X	X
(Saoud, 2002)		X			2002		X
(Cherif & Davidsson, 2010)		X			2010		X
(Agarwal & Umphress, 2010)		X			2010		X
(Martin & Raffo, 2001)	X		X	X	2001	X	X
(Ioana Rus et al., 1999)	X		X	X	1999		X
(SeungHun Park et al., 2008)			X	X	2008		X
(I. Rus et al., 2003)			X		2003	X	X
(H. Zhang et al., 2011)			X		2011		X
(Filho & Rocha, 2010)				X	2010	X	
(Rodríguez et al., 2011)				X	2011	X	
(Raffo et al., 1999)				X	1999	X	
(Seunghun Park & Bae, 2011)				X	2011	X	

Adicionalmente, como se puede ver en la Tabla 2, de los trabajos seleccionados, la mayoría fundamenta la Dinámica de Sistemas que es el paradigma escogido para orientar el proceso de modelado y simulación que se sigue en las próximas secciones.

Tabla 2 - Proporciones de los trabajos revisados, clasificados por paradigma de simulación.

Cantidad de trabajos referenciados: 28		**Proporción**	**% del total**
De los que:			
	De Dinámica de Sistemas:	14/28	50%
	De Agentes y Sistemas Multi-agente	7/28	25%
	De Simulación de Eventos Discretos	6/28	21%
	De otros enfoques	8/28	28%
Observación:	Existe solapamiento en las proporciones de los trabajos como consecuencia de los trabajos híbridos que pueden corresponderse a más de un paradigma de modelado y simulación.		

De importancia en este sentido resultan las conclusiones de (He Zhang et al., 2008), donde se argumenta, tras una extensa investigación, que el paradigma con mayor potencial de uso y aprovechamiento para modelado y simulación de procesos software es la Dinámica de Sistemas, aunque siempre teniendo en cuenta que dependiendo del caso en particular todos pueden ser útiles, y en particular los modelos híbridos comienzan a tomar importancia.

3 Propuesta

3.1 Método de Craig Larman

El método de Craig Larman es una instanciación concreta del *Rational Unified Process* (RUP), que busca de una manera clara dirigir el proceso de desarrollo de software manteniendo siempre de forma clara la trazabilidad del software. A continuación se presentan sus principales características y un bosquejo esquemático del método en su totalidad.

3.1.1 Características del Método de Larman:

- Es un método de desarrollo de software orientado a objetos.
- Tiene sus raíces en el Método RUP (*Rational Unified Process*).
- RUP es abierto, multi-caminos, y por ende más apropiado para desarrolladores senior.
- En el Método de Larman se toma uno de los caminos del RUP y se describe.
- Se ve claramente la traza de un paso a otro.
- De manera intuitiva lleva a la construcción del software.
- RUP es adaptativo, incremental, dirigido por los casos de uso. Adaptativo porque en función de las características del proyecto, se puede usar un ciclo de vida u otro.
- En el Método de Larman, a diferencia de RUP, hay un ÚNICO ciclo de vida a seguir y éste ciclo de vida es iterativo, incremental, y dirigido por los casos de uso.

Iterativo -> Cuando se dice que el método es iterativo, este está persiguiendo una filosofía de "Divide y vencerás". Sin dejar de tener una perspectiva sistémica y en la que se estudia el sistema en amplitud, éste es dividido en partes, y es abordado desde el principio hasta el final por partes. Se aplica desde el análisis hasta la codificación y

pruebas en pequeños tramos de funcionalidad: en pequeños subconjuntos de casos de uso.

Incremental -> es incremental en el sentido de que cada mini proyecto dentro del macro desarrollo va sobre el anterior. Siendo eso así, entonces se desarrolla un conjunto de casos de uso desde el principio hasta la codificación y las pruebas, y luego, el siguiente conjunto de casos de uso a codificar iría encima del anterior. El producto se va haciendo cada vez más grande permitiendo una disponibilidad del software en menos tiempo, de cara al cliente, quien percibe el incremento, la funcionalidad y operatividad.

Dirigido por casos de uso -> Los casos de uso son la pieza inicial con la que se va a trabajar. El caso de uso va sufriendo transformaciones, pasando por diagrama de clases, diagrama de secuencia, y así sucesivamente, hasta que se llega al código.

3.1.2 Etapas del Proceso definido por Larman.

Fundamentalmente, Larman conceptualiza su instanciación del *Rational Unified Process* en un proceso de tres macro-etapas o grandes fases:

- Planificación y Especificación de Requisitos.
- Construcción.
- Instalación.

3.1.3 Bosquejo conceptual del Método de Larman

El Método de Larman (Figura 9), al ser una versión simplificada del RUP, tiene la característica de que las fases de "Planificación y Especificación de Requisitos" e "Instalación" sólo se realizan una vez, mientras que la fase de "Construcción" es una fase compuesta ya que se compone de varios ciclos de desarrollo, sobre los cuales se agrupan casos de uso, y en los que a su vez, cada ciclo de desarrollo tiene una serie de actividades internas que se llevan a cabo, tales como: Diseño, Análisis, Implementación, Refinamiento del Plan y Sincronización del Modelo.

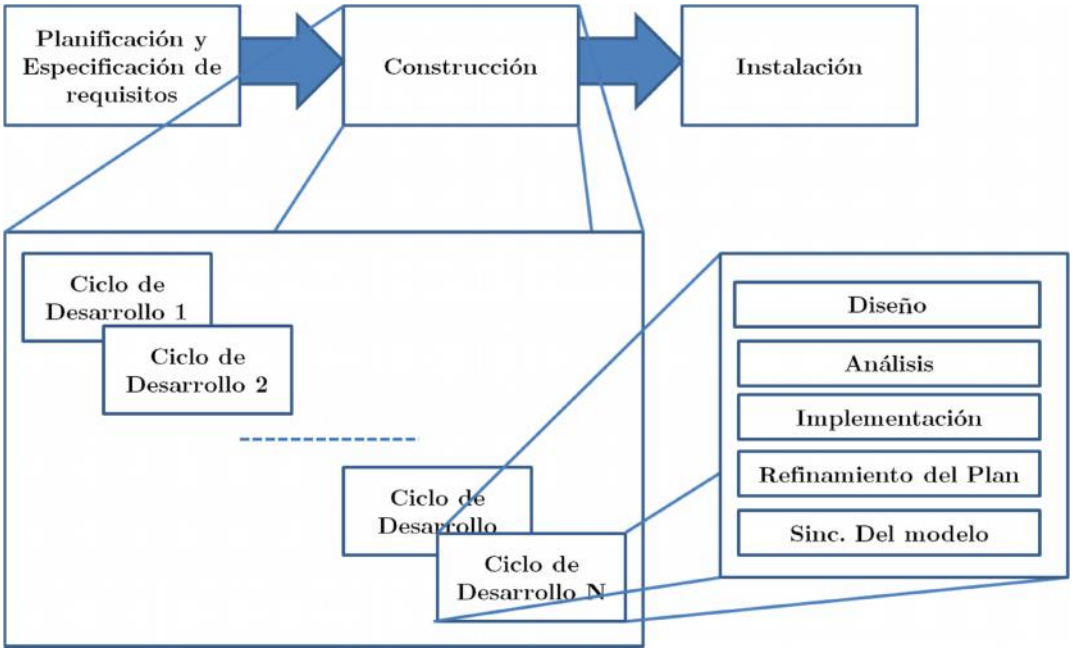

Figura 9 - Bosquejo conceptual del Método de Larman

3.2 Método de Larman y Dinámica de Sistemas, ¿Punto de encuentro?

En las siguientes secciones, se presentará el diseño y creación de un modelo de simulación del proceso software guiado por el método de Larman. El Método de Larman describe la naturaleza del sistema real a ser simulado, es decir, el Método de Larman describe el proceso de desarrollo de software que se va a simular. Por otro lado, la Dinámica de Sistemas proporciona todo el marco conceptual para la construcción del modelo; es así que la Dinámica de Sistemas invita a construir el modelo de simulación a partir de niveles o acumulaciones, flujos de material o tareas y ciclos de realimentación presentes e identificados en el sistema real, cruciales en el funcionamiento adecuado del modelo de simulación.

3.3 Software a utilizar: VENSIM

Existe a disposición de la comunidad académica una serie de paquetes informáticos que se pueden usar para el diseño y construcción de modelos de Dinámica de Sistemas. Entre los software disponibles se cuentan: DYNAMO, ITHINK, POWERSIM, STELLA y VENSIM. Se ha decidido trabajar con VENSIM 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. Para la comunidad empresarial, existe una versión de uso bajo pago de licencia, que autoriza su explotación para fines que puedan derivar en beneficios económicos como consecuencia de su uso.

El software VENSIM permite representar los elementos que caracterizan el sistema a modelar (niveles, flujos, variables auxiliares, constantes del sistema, etc.), las relaciones entre estos elementos, y permite establecer los valores numéricos de las variables del sistema, por medio del uso e implementación de funciones y tablas[8]. El software VENSIM, además de permitir la representación del sistema real, es fácilmente manipulable y modificable para experimentar con variaciones de los parámetros usados en el modelo, resultando así en 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 (García, 2006).

3.4 Selección de Parámetros del Proceso Software

Para poder construir el modelo de simulación de un proceso de Ingeniería del Software usando un lenguaje de Dinámica de Sistemas, es necesario contar con la estimación de algunos parámetros que intervienen en el proceso de ingeniería del software. En el área de la Ingeniería del Software, se ha hecho uso de una herramienta para estimar

[8] Las funciones permiten experimentar con valores a partir de funciones matemáticas conocidas, mientras que las tablas permiten con valores a partir de tablas (o gráficas) de pares de valores x e y, que suelen ser observaciones u estimaciones inexactas de comportamientos en el sistema real.

parámetros del proceso de Ingeniería del Software que ha resultado ser de mucha utilidad: La herramienta de estimación COCOMO, y su variante COCOMO II. Las herramientas COCOMO son el resultado de un amplio y arduo trabajo de investigación en el que se identificaron una serie de parámetros que influyen y son determinantes en la estimación de proyectos de desarrollo de software, y por consiguiente son parámetros que afectan al proceso de desarrollo de software. En el modelo de simulación (en VENSIM) del proceso software que se construirá en este trabajo, se hará uso de los parámetros que el modelo COCOMO ofrece, por lo que a continuación se presenta una breve introducción a esta herramienta para luego pasar a justificar los parámetros que se escogen para el diseño de modelos dinámicos del proceso software.

3.4.1 COCOMO® II: una referencia de estimación a partir de parámetros.

La herramienta para estimación de esfuerzo y costes denominada COCOMO® II (*Constructive Cost Model II*), es una versión más completa y mejorada de la herramienta COCOMO® (*Constructive Cost Model*). La herramienta, es 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 (California., 2003), cuyos principales objetivos son:

- Proveer a usuarios, administradores, e ingenieros de software y de sistemas, de una herramienta que permita estimar costes y planificación del desarrollo de software orientado hacia las prácticas de ciclos de vida de los procesos de desarrollo.
- Proporcionar vías de mejora continua al proveer del desarrollo de una base de conocimiento de costes de desarrollo.
- Proveer desarrolladores de un marco analítico y un conjunto de herramientas y técnicas para la evaluación de las mejoras tecnológicas del software con respecto a los ciclos de vida del software y su planificación.

De acuerdo con sus creadores, la herramienta COCOMO® II permite la estimación de coste, esfuerzo y programación de tareas en el proceso de planificación de un nuevo

desarrollo de software. Este modelo de estimación permite la posible 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:

- **Atributos de Software:** para representar las características de las propiedades de un proyecto referentes a los activos de software disponibles.
- **Atributos de Hardware:** para representar las limitaciones y características del equipamiento informático disponible.
- **Atributos de Personal:** para representar las características de las personas que intervienen en el proceso.
- **Atributos del Proyecto:** para representar las características del proyecto en sí en cuanto a capacidades y restricciones de recursos.

Una descripción detallada de los atributos que toma en cuenta el COCOMO II, las funciones que usa y como se interrelacionan se encuentra disponible en (Boehm, 2012). El listado de atributos por tipo se presenta en la Tabla 3.

Para una mejor comprensión, a continuación se va a dar una breve explicación de cada uno de los atributos de acuerdo a la clasificación que se acaba de mostrar en la Tabla 3, y, en Tabla 4, Tabla 5, Tabla 6 y Tabla 7, se presentan los valores descriptivos válidos para cada uno de los atributos, clasificados por tipo; y en la Tabla 8 se presenta el rango de valores con que todos estos atributos son manejados en el modelo específico de COCOMO II:

Tipo de atributo:	Atributo	Acrónimo
Atributos de Software	• Fiabilidad • Tamaño de Base de Datos • Complejidad	• RELY • DATA • CPLX
Atributos de Hardware	• Restricciones de tiempo de ejecución • Restricciones de memoria virtual • Volatilidad de la máquina virtual • Tiempo de respuesta	• TIME • STOR • VIRT • TURN
Atributos de Personal	• Capacidad de análisis • Experiencia en la aplicación • Calidad de los programadores • Experiencia en la máquina virtual • Experiencia en el lenguaje	• ACAP • AEXP • PCAP • VEXP • LEXP
Atributos de Proyecto	• Técnicas actualizadas de programación • Utilización de herramientas de software • Restricciones de tiempos de desarrollo	• MODP • TOOL • SCED

Tabla 3 - COCOMO II: Atributos por categoría

3.4.2 Atributos de Software

Fiabilidad (RELY): se trata de la garantía de funcionamiento requerida al software, y es un indicador de las posibles consecuencias para el usuario de los defectos que el producto software pueda tener. Valores "muy bajos" indicarían que es necesario eliminar los defectos, pero que esto no tiene mayor consecuencia; valores "bajos" implican que hay riesgo de pérdidas pero que éstas son fácilmente recuperables por los usuarios; valores "nominales" implican riesgo de pérdidas para los usuarios pero que esta situación es resoluble sin mayor dificultad; valores "altos" representan pérdidas monetarias significativas o inconveniencias humanas, mientras que valores "muy altos" implican la pérdida de vidas humanas.

Tamaño de Base de datos (DATA): es un indicador del tamaño de la base de datos a desarrollar en relación con el tamaño del programa.

Complejidad (CPLX): es un indicador integral de la complejidad de los módulos y el sistema que el producto representa. Puede variar de "muy bajo" en los casos en que sólo se requieren expresiones matemáticas simples a "extremadamente alto" para casos de consumo excesivo de recursos de planificación.

COCOMO II: Parámetros y atributos		Valor descriptivo del parámetro					
ID:	**Atributo de Software**	**Muy bajo**	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra alto**
RELY	**Fiabilidad**	Un fallo implica simple corrección.	Un fallo implica una perdida fácilmente recuperable.	Un fallo implica trabajo y pérdida pero se puede corregir.	Un fallo implica gran pérdida financiera o humana.	Un fallo implica pérdida de vidas humanas.	___
DATA	**Tamaño de Base de Datos (Puntuación)**	___	10	100	1000	___	___
CPLX	**Complejidad**	Sólo lo hacen complejo ciertas expresiones matemáticas simples	Las expresiones matemáticas no son tan triviales y requieren cierto recurso.	Las expresiones implícitas y recursos requeridos son habituales.	Se requiere de algunas pocas intervenciones de recursos especiales.	Se requiere de muchos recursos y las expresiones son complejas.	Se requieren muchos recursos de planificación

Tabla 4 - Explicación de los valores de Atributos del Software de un Proyecto.

3.4.3 Atributos de Hardware:

Restricción de tiempo de ejecución (TIME): es un indicador de las exigencias para los programadores de que el programa se ejecute bajo ciertas restricciones de tiempo. Va desde "nominal" cuando el porcentaje de tiempo de ejecución disponible es del 50% hasta "extremadamente alto" cuando la restricción es del 95%.

Restricciones de memoria virtual (STOR): se refiere a restricciones de almacenamiento principal, y va desde "nominal" cuando la restricción de almacenamiento es del 50% a "extremadamente alto" cuando es del 95%. Este atributo es una representación de cómo se ve incrementado el esfuerzo de programación cuando se ve limitado el acceso a capacidad de almacenamiento principal.

Volatilidad de la máquina virtual (VIRT): Es una característica propia del equipo en el que se va a desarrollar y como la palabra "volatilidad" nos llevaría a pensar, hace referencia a la susceptibilidad de este equipo para sufrir cambios. Puede ir desde "bajo" para equipos poco volátiles hasta "muy alto" para equipos con un alto grado de posibilidad de ocurrencia de cambios.

Tiempo de respuesta (TURN): Este indicador representa una forma de medir el tiempo de respuesta que tiene el ordenador desde la perspectiva del equipo de desarrollo. Cuanto mayor es el tiempo de respuesta, más alto es el esfuerzo humano implícito; es decir: son directamente proporcionales. Este tiempo puede ser desde "bajo" para sistemas muy interactivos hasta "muy alto" para tiempos de respuesta mayores a las doce horas.

COCOMO II: Parámetros y atributos		Valor descriptivo del parámetro					
ID:	**Atributo de Hardware**	**Muy bajo**	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra alto**
TIME	Restricciones de tiempo de ejecución	___	___	50%	…..	…..	95%
STOR	Restricciones de memoria virtual	___	10	100	1000	___	___
VIRT	Volatilidad de la máquina virtual	___	Poco volátil	…..	…..	Muy volátil	___
TURN	Tiempo de respuesta	___	Sistema interactivo	…..	…..	Respuesta demora >12 horas	___

Tabla 5 - Explicación de los valores de Atributos del Hardware de un Proyecto

3.4.4 Atributos de Personal

Capacidad de análisis (ACAP): se refiere cuán capacitado está el equipo de analistas con respecto a habilidades de análisis, eficiencia y cooperación. Es uno de los términos más complejos que se introducen en el modelo de COCOMO y en el área de la Ingeniería de Software ya que no es fácilmente cuantificable y sus mediciones suelen estar sujetas a críticas y/o subjetividades. Tiene un alto impacto en el esfuerzo humano que el sistema calcula, de manera que cuanto mayor es la capacidad de análisis del

equipo, menor es el esfuerzo necesario. Su escala es desde “muy bajo” para equipos muy poco capacitados hasta “muy alto” para equipos con buena capacidad de análisis.

Experiencia en la aplicación (AEXP): Se refiere a la experiencia que el equipo tiene en trabajos relacionados con aplicaciones similares a la que se esté desarrollando. Esta experiencia tiene una gran influencia en el esfuerzo y puede ir desde “muy baja” para los casos de menos de cuatro meses de experiencia, hasta “muy alto” para el caso de equipos con más de doce años de experiencia.

Calidad de los programadores (PCAP): en este caso, similar al caso de los analistas, se mide la capacidad de los programadores para llevar a cabo sus tareas. Se trata de una medición grupal de las habilidades de programación que tiene el equipo de programación y su aprovechamiento.

Experiencia en la máquina virtual (VEXP): en este caso se hace referencia a la experiencia con el procesador que el equipo tiene. Existe una relación inversamente proporcional entre esta experiencia y el esfuerzo requerido para un proyecto de desarrollo, ya que cuanto mayor es la experiencia del grupo, menor es el esfuerzo necesario. Tiene una escala que varía desde “muy bajo” para el caso de experiencia menor a un mes, hasta “alto” cuando esta experiencia es mayor a tres años.

Experiencia en el lenguaje (LEXP): A pesar de que los lenguajes de programación tienen cierta tendencia a universalizarse y de que la lógica implícita en el conocimiento de un lenguaje es un fuerte catalizador al momento de aprender otro lenguaje, es un hecho aceptable que cuanto mayor sea la experiencia que un grupo de programadores tenga con un lenguaje determinado, pues su trabajo de codificación será mucho más seguro, menos tendente a errores y por ende implicará un menor esfuerzo humano. Su escala varía desde “muy bajo” hasta “alto” para grupos que se presenten desde un mes hasta tres años de experiencia respectivamente.

COCOMO II: Parámetros y atributos		Valor descriptivo del parámetro					
ID:	**Atributo de Personal**	**Muy bajo**	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra alto**
ACAP	Capacidad de análisis	El equipo está muy poco capacitado	___	…..	…..	El equipo supera todas las expectativas sobre sus capacidades	___
AEXP	Experiencia en la aplicación (media)	≤ 4 meses	1 año	3 años	6 años	≥ 12 años	___
PCAP	Calidad de los programadores	El equipo está muy poco capacitado	…..	…..	…..	El equipo supera todas las expectativas sobre sus capacidades	___
VEXP	Experiencia en la máquina virtual	≤ 1 mes	4 meses	1 año	≥ 3 años	___	___
LEXP	Experiencia en el lenguaje	≤ 1 mes	4 meses	1 año	≥ 3 años	___	___

Tabla 6 - Explicación de los valores de Atributos del Personal de un Proyecto

3.4.5 Atributos del Proyecto

Técnicas actualizadas de programación (MODP): Es una representación del ajuste del equipo de desarrollo al uso de técnicas y prácticas modernas de programación. Tiene una escala que va desde "muy bajo" para el caso en que no se utilizan técnicas modernas y experimentadas de programación, hasta "muy alto" para el caso en que ya es habitual el uso de técnicas modernas de programación.

Utilización de herramientas de software (TOOL): en este caso lo que se observa es si se da un uso adecuado a las herramientas software, ya que estas son multiplicadores de la productividad del equipo (con lo que el esfuerzo sería menor). Tiene una escala que va desde "muy bajo para el caso en que el uso es muy básico, hasta "muy alto" para el caso en que el uso correcto de estas herramientas es una práctica ya implantada en la cultura organizativa del equipo.

Restricciones de tiempos de desarrollo (SCED): Cuando nos referimos al tiempo de desarrollo de un producto software, hay que hacer notar que el tiempo nominal (o

natural) es el que menor esfuerzo implica. Si se aplican demoras o aceleraciones a este tiempo de desarrollo, siempre esto implicará un mayor esfuerzo.

COCOMO II: Parámetros y atributos		Valor descriptivo del parámetro					
ID:	**Atributo de Proyecto**	**Muy bajo**	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra alto**
MODP	Técnicas actualizadas de programación	No hacen uso de prácticas modernas.	El uso de las prácticas modernas es experimental	Hay alguna experiencia razonable usando técnicas modernas.	Experiencia extendida en uso de técnicas modernas	Uso habitual de técnicas modernas.	___
TOOL	Utilización de herramientas de software	Sólo se utilizan herramientas básicas.	…..	…..	…..	Se usan herramientas específicas.	___
SCED	Restricciones de tiempos de desarrollo	Apresuramiento	…..	Curso normal	…..	Retraso	___

Tabla 7 - Explicación de los valores de Atributos propios de un Proyecto

En COCOMO II, para hacer uso y extrapolar toda la información sobre un proyecto, haciendo uso de los quince atributos explicados anteriormente, diseñaron una escala numeral que aporta los valores de entrada a los modelos y fórmulas matemáticas de las que hace uso para estimar el coste y esfuerzo de un proyecto. A continuación, en la Tabla 8, se observa los valores asignados para cada uno de los parámetros, lo que, de momento, nos da una idea de una forma de formalizar y parametrizar atributos observables y presentarlos de una manera cuantificable.

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	___

Tabla 8 - Parámetros-Atributos del modelo de Estimación de costes COCOMO II.

3.4.6 Interfaz de gestión en el USC-COCOMO II:

La herramienta lo que permite es realizar la estimación de recursos haciendo uso de una interfaz en la que el usuario tiene la posibilidad de configurar los parámetros según el caso que le interesa (Figura 11) y luego observar de una manera concisa (Figura 10) los valores que le son útiles en la gestión de un proyecto software.

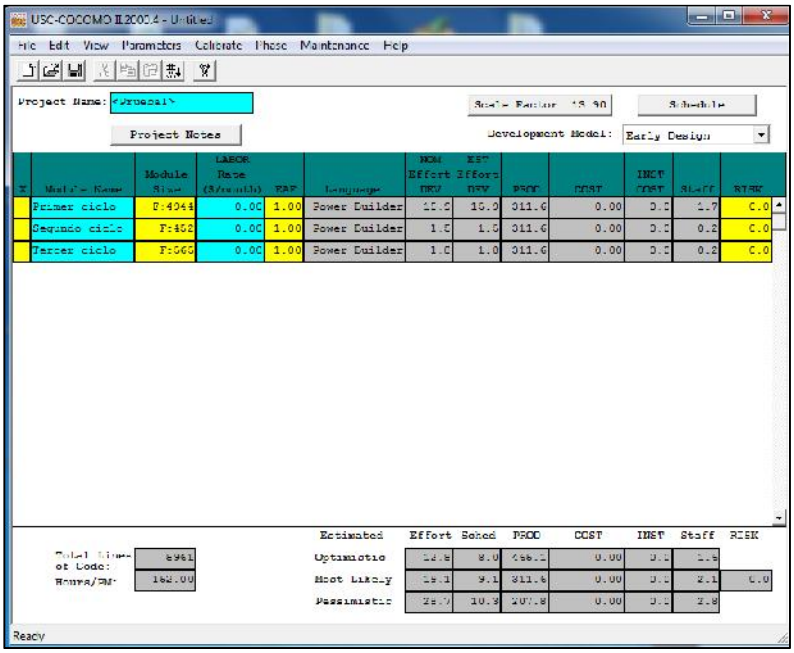

Figura 10 - Interfaz del software para estimación COCOMO II

3.4.7 Atributos de desarrollo configurables en la opción "Diseño temprano":

La opción de "Diseño temprano" permite configurar siete atributos de desarrollo: RCPX, RUSE, PDIF, PERS, PREX, FCIL, SCED, y sus correspondientes escalas y valores.

Tabla 9 - Parámetros COCOMO del modelo Early Design

Acrónimo	Parámetro	Parámetros del modelo Post-architecture que incorpora:
RCPX	Fiabilidad y complejidad del producto	Combina los parámetros RELY, DATA, CPLX y DOCU. Su escala es desde 5 hasta 21. Ver Tabla 10.
RUSE	Reutilización requerida	Se corresponde con el del modelo Post-architecture.
PDIF	Dificultad de la plataforma	Combina los parámetros TIME, STOR y PVOL. Su rango de valores va desde 8 hasta 17. Ver Tabla 10.
PERS	Capacitación del personal	Combina los parámetros ACAP, PCAP, y PCON[9], Cada uno de los que es puntuable entre 1 (Muy bajo) y 5 (Muy alto), con lo que PERS varía entre 3 y 15.El valor nominal es 9 (3+3+3). Ver Tabla 10.
PREX	Experiencia del personal	Combina AEXP, PEXP, Y LEXP. Su rango de valores va de 3 a 15.
FCIL	Instalaciones	Combina TOOL y SITE. Varía desde 2 hasta 11.
SCED	Programación/Planificación	Se corresponde con el del modelo Post-architecture.

[9] Explicados desde la sección 3.4.2 hasta la 3.4.5.

Los rangos de posibles valores para los parámetros del modelo *Early-design* se presentan resumidos a continuación en la Tabla 10:

Tabla 10 - Rango de valores de parámetros del modelo Early-design

Parámetro	**Correspondencia**	**Escala**						
PERS	Capacitación del personal	Extra-bajo	Muy-Bajo	Bajo	Nominal	Alto	Muy-Alto	Extra-Alto
		3,4	5,6	7,8	9	10,11	12,13	14,15
RCPX	Fiabilidad y complejidad del producto	Bajo	Bajo				Alto	Alto
		5,6	7,8	9 a 11	12	13 a 15	16 a 18	19 a 21
RUSE	Reutilización requerida		Muy-Bajo	Bajo	Nominal	Alto	Muy-Alto	Extra-Alto
				Ninguno	Durante el Proyecto	Durante el programa	Durante la línea de productos	Durante múltiples líneas de
PDIF	Dificultad de la plataforma			Bajo	Nominal	Alto	Muy-alto	Extra-alto
				8	9	10 a 12	13 a 15	16,17
PREX	Experiencia del personal	Extra-bajo	Muy-Bajo	Bajo	Nominal	Alto	Muy-Alto	Extra-Alto
		3,4	5,6	7,8	9	10,11	12,13	14,15
FCIL	Instalaciones	Extra-bajo	Muy-Bajo	Bajo	Nominal	Alto	Muy-Alto	Extra-Alto
		2	3	4,5	6	7,8	9,1	11
SCED	Programación / Planificación		Muy-Bajo	Bajo	Nominal	Alto	Muy-Alto	Extra-Alto
			75%* Nominal	85%	100%	130%	160%	

En la herramienta COCOMO, estos parámetros son configurables por medio de una ventana en la que se pueden asignar los valores según sea el caso que nos interese. Ejemplo de esto es la Figura 11 que se presenta a continuación.

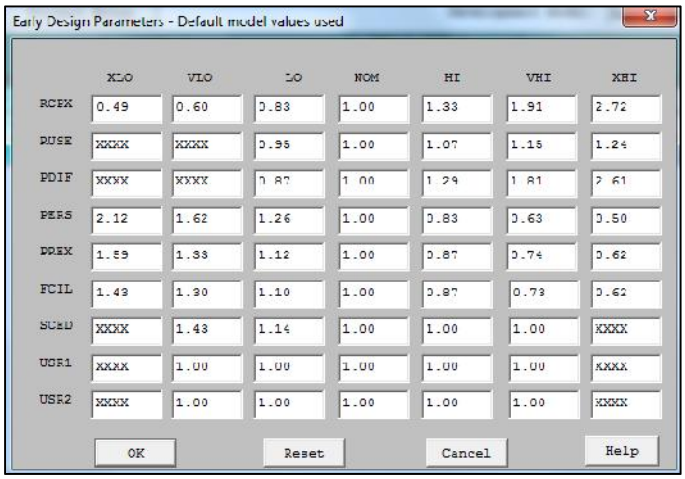

Figura 11 - Parámetros y valores de entrada que considera COCOMO II

Es la opción más completa, y la que mayor aceptación y uso ha tenido, permite trabajar de un modo más cercano a la realidad al proporcionar valores y estimaciones más reales. A efectos de este trabajo, nos interesa conocer el significado y uso de estos parámetros y la forma de incorporarlos en el modelo a construir. El detalle de estos parámetros está en la sección 3.4.1.

3.5 Construcción del modelo

Se usará el software de simulación Vensim (Ventana Systems, 2011), para construir bajo el enfoque de la Dinámica de Sistemas, un modelo de simulación de un proceso de Ingeniería del Software, regido por el Método de Craig Larman. Teniendo como referencia la metodología de (Sterman, 2000) para modelado con Dinámica de Sistemas, primero se hará 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 se hará la configuración del modelo en el lenguaje del software usado (Vensim), para su

posterior uso, evaluación, y análisis de funcionalidad. A continuación, la Figura 12 representa un resumen de los pasos seguidos en esta sección:

Figura 12 - Metodología de Sterman para Dinámica de Sistemas, llevado al caso del Método de Larman

3.5.1 Descripción del sistema a modelar: *El Método de Craig Larman*

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, como se pudo ver en la sección 3.1. Sin embargo, el funcionamiento de este conjunto de elementos va más allá de esa mera descripción, resultando realmente en un comportamiento dinámico y lleno de causalidades variantes en el tiempo. A continuación se presenta un modelo de simulación de un proceso generalizado de desarrollo de software dirigido por casos de uso, como lo describe el Método de Craig Larman.

Como una primera aproximación al proceso de desarrollo de software que describe el Método de Craig Larman, es necesario comprender que el proceso en sí se compone de tres fases o macro etapas claramente diferenciadas una de otra. Estas etapas, esencialmente comunes a la gran mayoría de modelos y métodos de desarrollo de software, son:

- Planificación y Especificación de requisitos.
- Construcción.
- Instalación.

La fase de Planificación y Especificación de Requisitos, se ejecuta formalmente una sola vez durante todo el proceso de software y comprende en esencia la conceptualización, especificación y documentación de toda la información concerniente al producto software que se quiere construir, con un alto grado de detalle, en que se definen claramente los objetivos, requisitos, y finalmente los casos de uso iniciales que tiene el proyecto de desarrollo de software.

La fase de Construcción, como su nombre indica, comprende todo el proceso durante el cual a partir de la planificación y especificación de requisitos, se ponen en marcha los recursos de personal y tecnológicos con el fin conseguir un producto software entregable al usuario. Durante la fase de construcción ocurre una dinámica continua entre todos los elementos intervinientes: equipos (de análisis, diseño, desarrollo, y pruebas), recursos (de software, de personas, de tiempo, económicos), el entorno o ambiente de desarrollo con todos sus factores y variantes.

La fase de Instalación comprende la entrega, puesta en marcha, demostración y entrenamiento del producto en funcionamiento y con un alto grado de calidad al cliente. Suele ser la fase que cierra el proceso general de desarrollo del producto, tomando en cuenta que siempre se dan tareas adicionales de mantenimiento y reingeniería.

3.5.2 Delimitación del modelo a construir

Tomando en cuenta que el dinamismo de todo el proceso de desarrollo de software de Larman no es equitativo entre las tres fases que lo componen, sino que la gran mayoría de la actividad e interactividad se da en la fase de construcción, se ha decidido limitar a esta fase el proceso a ser modelado y simulado a efectos de este trabajo.

3.5.3 Diagrama causal del modelo.

En un diagrama causal se representan todas las causalidades que se presentan entre los elementos que componen un sistema o proceso. A continuación, como si se tratase de una especie de "zoom" o "acercamiento" hacia lo que realmente queremos representar, veremos 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 veremos 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; y 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, en términos de niveles y flujos como los explicados en la sección 2.2.

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

Las tres fases del proceso, lejos de estar desconectadas y ser fases de un desarrollo tipo cascada, en que una etapa es totalmente completada antes de avanzar a la siguiente, están interconectadas, y, como se observa en Figura 9 y Figura 13, suelen tener una fuerte interacción durante todo el proceso de desarrollo de software. 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-make*" de actividades que corresponden a la fase de construcción. 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.

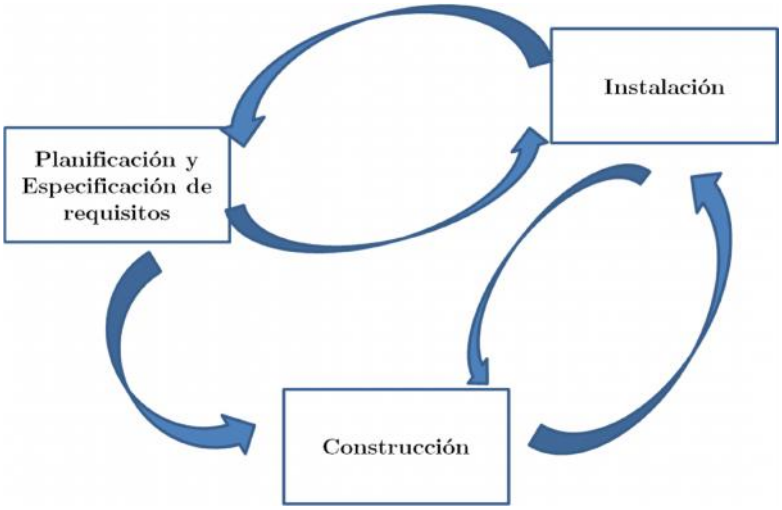

Figura 13 - Diagrama general de fases del Método de Larman

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

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 sub-etapas de esta fase. Estas sub-etapas 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. Como se puede ver en la Figura 14, los ciclos de desarrollo están relacionados entre sí y son directa pero no estrictamente proporcionales a la cantidad de casos de uso que el proyecto tenga y la complejidad de éstos. 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. 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 14, 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 14, 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.

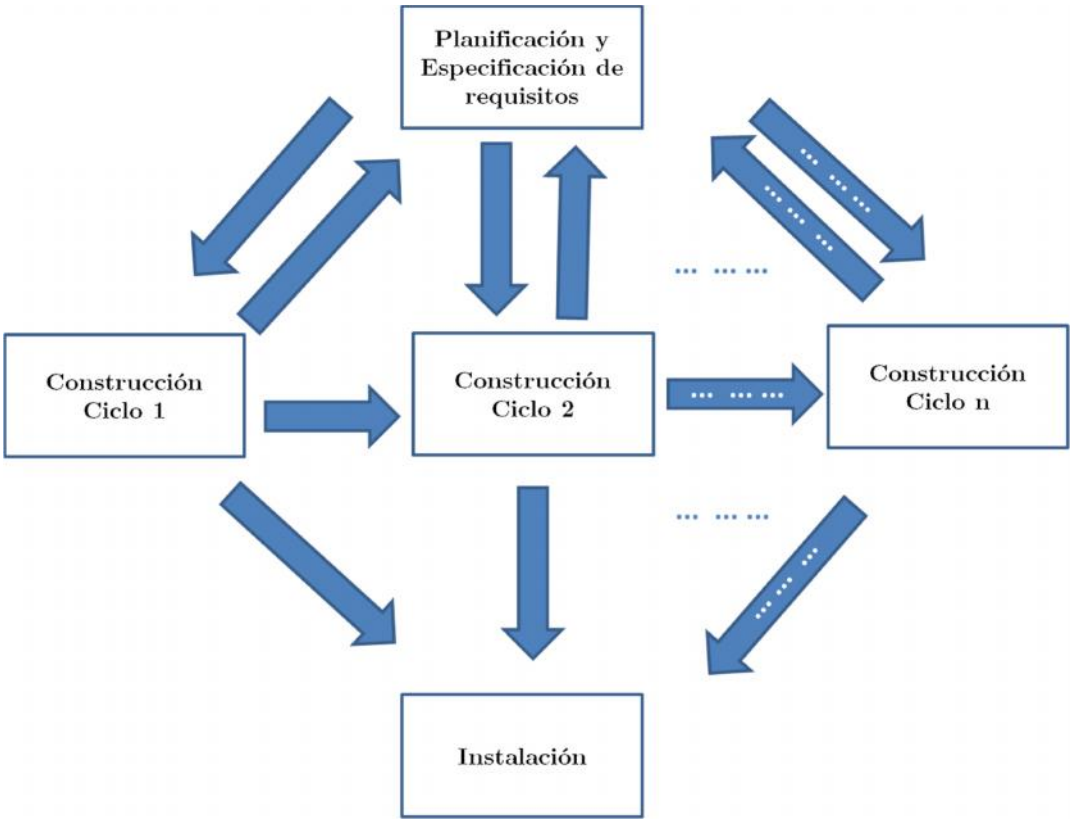

Figura 14 - Primer desdoblamiento de la complejidad del proceso: Construcción

Diagrama en detalle: Fase de construcción

A continuación, 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 han agregado la

mayor cantidad de elementos posibles con la finalidad de hacer la representación pictográfica más representativa.

Siguiendo el sentido de las flechas azules, 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 (Sterman, 2000) justifica el uso de la Dinámica de Sistemas para modelar sistemas como éste.

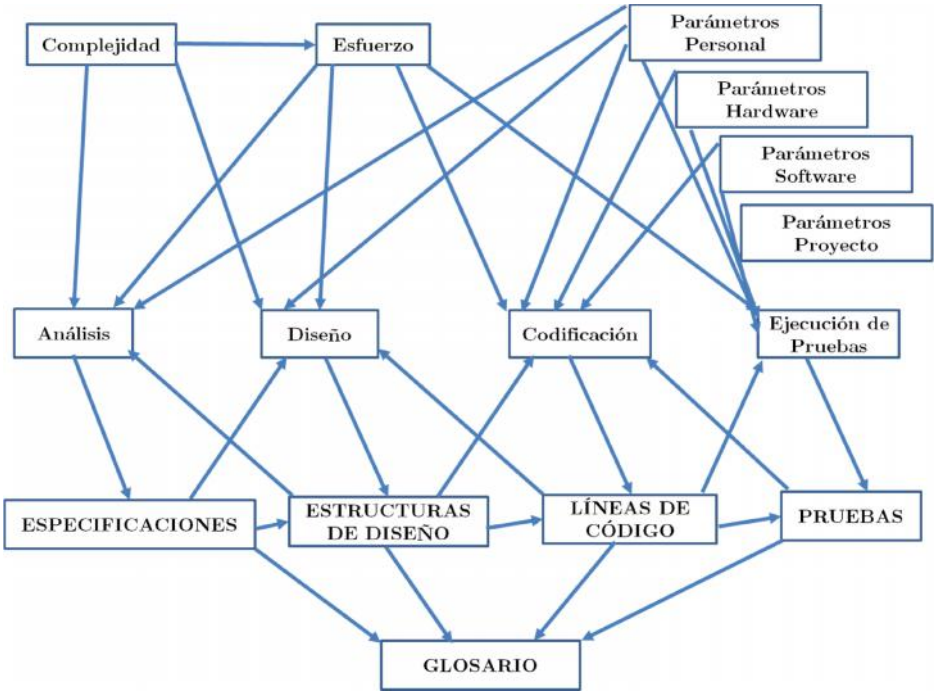

Figura 15 - Diagrama causal de la fase "Construcción" del Método Larman de ingeniería del Software

3.5.4 Identificación de flujos, niveles, variables, constantes, elementos auxiliares y ciclos de realimentación del modelo.

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.

Las especificaciones están constituidas por todas las descripciones detalladas y extendidas que el equipo de Análisis presenta como consecuencia de haber estudiado a fondo el dominio del problema y haber identificado y detallado los casos de uso del diseño que se está construyendo. En esta fase, más enfocada en explicar el *qué* que el *cómo* del sistema, las especificaciones son logradas a medida que la comprensión del problema alcanza una sólida madurez. Entre las tareas especificadas por el Método de Larman que contribuyen a conseguir las especificaciones del Análisis se cuentan:

- Definición de los casos de uso en formato expandido.
- Refinamiento de los Diagramas de Casos de Uso.
- Refinamiento del Modelo Conceptual del sistema.
- Refinamiento del Glosario.
- Definición de los Diagramas de Secuencia del Sistema.
- Definición de los Contratos de operación.
- Definición de los Diagramas de Estado.

El comportamiento esperado del nivel "Especificaciones" tiene su mayor actividad en la etapa inicial del proyecto, aunque se mantienen ciertas actividades de análisis durante todo el proceso pero sin ser tan fuertes como en la etapa inicial.

Estructuras de Diseño (Diseños).

Las estructuras de diseño, o *diseños*, corresponden a soluciones a nivel lógico para satisfacer los requisitos del sistema. El *Método de Larman* especifica la fase de diseño como aquella en la que se diseña cómo se va a comportar el sistema para llevar a cabo las funcionen que le son solicitadas y especificadas en la fase de análisis y en la especificación de requisitos.

Entre las actividades que contribuyen a alcanzar la madurez de la fase de diseño y por ende una mayor cantidad de estructuras y especificaciones de diseño se cuentan:

- Definición de los casos de uso reales.
- Definición de Informes e Interfaz de Usuario.

- Refinamiento de la Arquitectura del Sistema.
- Definición de los Diagramas de Interacción.
- Definición del Diagrama de Clases de Diseño.
- Definición del Esquema de Bases de Datos.

El comportamiento esperado del nivel "Estructura de Diseño" es de mayor actividad en las etapas iniciales y media del proceso de construcción es general. Esto debido a que en las etapas consiguientes el énfasis de la actividad está en Implementación y pruebas.

Código.

El código es el principal producto de la fase de implementación del Método de Larman. En la implementación del sistema, todo lo que se haya especificado en la fase de Diseño es llevado a un lenguaje de programación, generando finalmente el(los) código(s) y sus correspondientes Diagramas de Componentes y de Despliegue (o Implantación).

El comportamiento esperado del código está dado por un crecimiento bastante lento al principio, luego acelerado y constante en la fase media de producción y hacia el final desacelerado y disminuyendo con aquellas excepciones en que haya re-codificación.

Pruebas realizadas.

Las pruebas son actividades que se realizan en todos los ciclos de desarrollo que se den en el proceso de construcción del software. Son determinantes dado que dependiendo del grado de superación y satisfacción de las pruebas, el proceso de ingeniería en general avanza o se mantiene en la fase actual.

El comportamiento esperado de las pruebas está dado por cierta actividad periódica ya que tienen su mayor vida cuando cada ciclo de desarrollo está culminándose.

Glosario.

El glosario es un *nivel,* dado que lo constituye una acumulación de palabras y sus especificaciones. Estas palabras y términos surgen durante todo el proceso de construcción aunque el mayor énfasis está en la fase de análisis y diseño, por lo que es así también el comportamiento esperado.

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.

Se ha llamado análisis al flujo de especificaciones que se producen durante toda la fase de construcción. A lo largo del tiempo, el análisis (correspondiéndose con la homónima etapa del diseño de Larman) conlleva una serie de actividades que en las etapas iniciales del proceso de construcción producen especificaciones que luego serán la base para las etapas posteriores de diseño, implementación, pruebas e implantación.

Diseño.

Correspondiéndose con la etapa homónima de la fase de construcción del Método de Larman, el flujo de diseño aquí presentado representa la producción de activos previos a la codificación del software. Este flujo es estrictamente dependiente de la existencia de las especificaciones que se producen en la etapa de Análisis y es fundamental para que la etapa de implementación se lleve a cabo con éxito.

Implementación.

Este flujo corresponde a la producción de código fuente a partir del diseño logrado en la etapa anterior. Aunque su actividad se concentra en la mitad del proceso general de construcción de software, puede darse el caso de que hay que recodificar luego de algunas pruebas (suele ser bastante común).

Corrección.

El flujo de corrección se entiende acá como el código que es borrado o eliminado como consecuencia de mejoras o correcciones al código general.

Pruebas.

El flujo de pruebas son las pruebas a que el código generado en la fase de implementación es sometido. Tiene fuerte actividad en la etapa final de cada ciclo de

desarrollo ya que la finalidad de las pruebas es validar el producto de las etapas anteriores del proceso de construcción.

3.5.5 Modelo en Vensim.

A continuación se presentará formalmente lo que es el modelo construido en este trabajo para el proceso de desarrollo de software del Método de Larman. Inicialmente, sin haber construido aún el contexto necesario para poder tomar decisiones sobre la investigación, se pensó en la construcción de un modelo general que englobara las tres fases del Método de Larman, con lo que surgió como inicio de la investigación el sub-modelo de la fase de *Planificación y Especificación de Requisitos* que se muestra en la Figura 16.

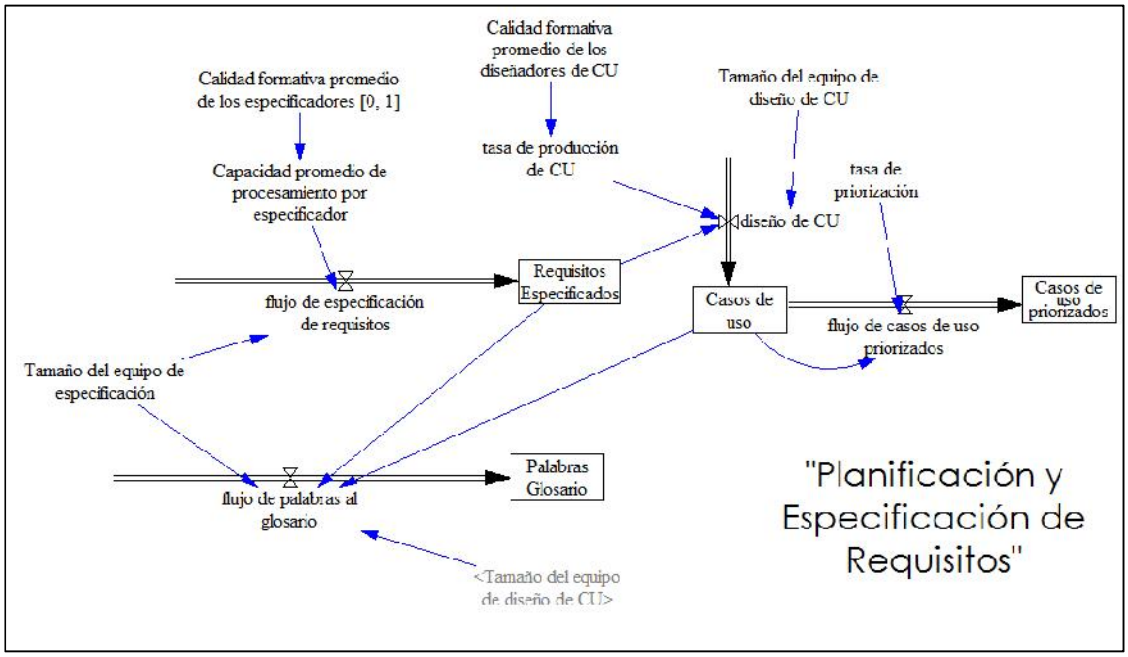

Figura 16 - Sub-modelo de simulación para la fase "Planificación y Especificación de Requisitos"

Sin embargo, fue necesario redefinir los objetivos del proyecto argumentado en la naturaleza del Método de Larman, en que las fases de "Planificación y Especificación de Requisitos" e "Instalación" sólo se llevan a cabo una vez, e implican tareas no muy dinámicas sino más bien claramente establecidas; mientras que la fase de "Construcción" tiene implícita toda una dinámica en sí misma.

La Dinámica de Sistemas es sumamente útil modelando sistemas dinámicos y complejos. La fase de construcción del Método de Larman, con todas las iteraciones, ciclos de desarrollo, dependencias y factores internos y externos que involucra, resulta idónea para ser modelada y simulada bajo este paradigma; por lo que a continuación se presenta el modelo de simulación para esta fase, teniendo en cuenta el deseo de dar continuidad a esta investigación con modelos mayores dentro de este mismo paradigma y otros alternativos bajo enfoques distintos.

El modelo de simulación

El modelo de simulación desarrollado está compuesto por tres *vistas* de la herramienta VENSIM. Cada una de estas vistas encierra información relevante del modelo, su configuración y funcionamiento.

La primera vista, presenta de forma esquemática los parámetros del proceso que afectan el proceso software y que fueron explicados en la sección 3.4. 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 los valores de los parámetros. En esta primera vista, presentada en la , también se ha incorporado la parte del modelo en la que se implementan las ecuaciones usadas por (Boehm, 2012) para la estimación de esfuerzo.

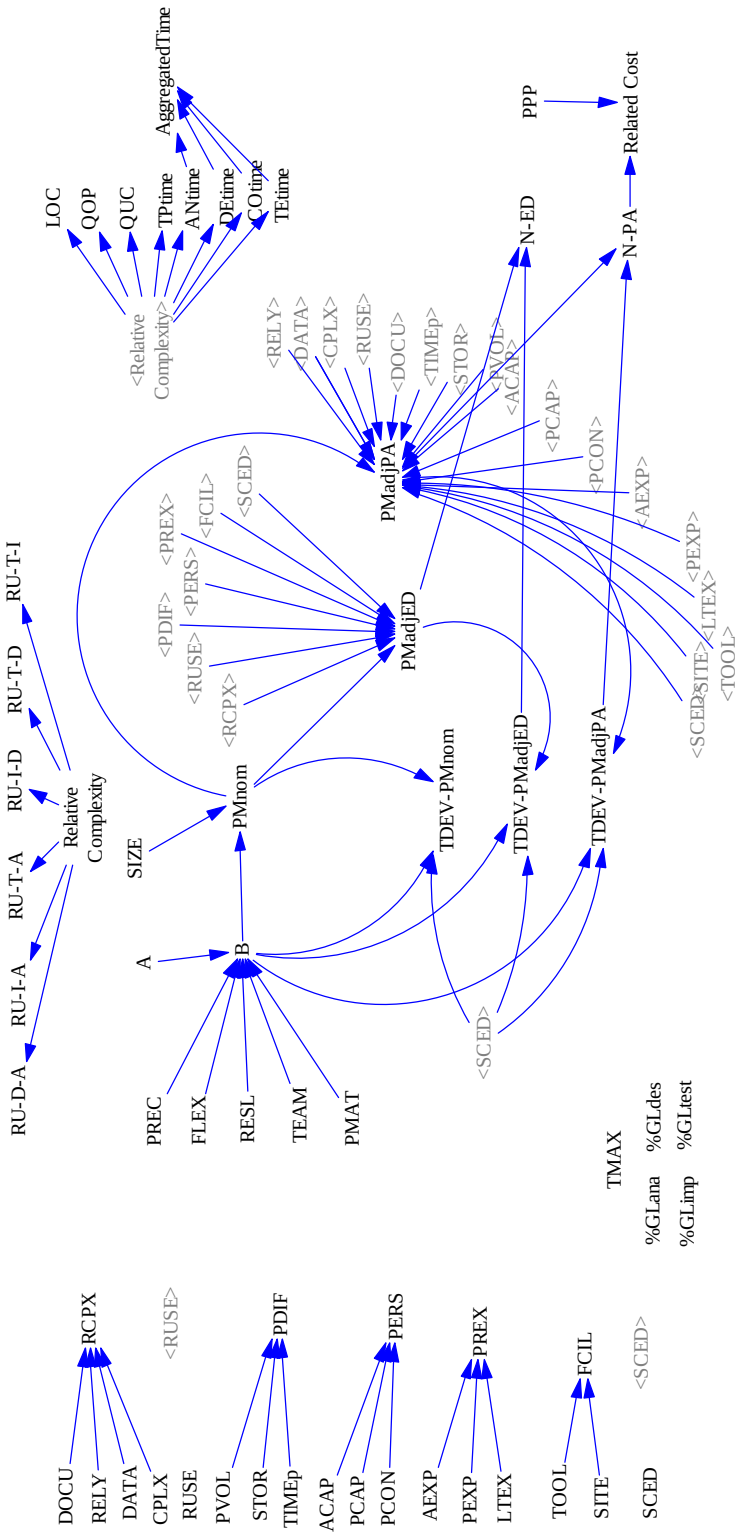

Figura 17 - Vista 1: Sub-modelo de configuración de parámetros y estimación.

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

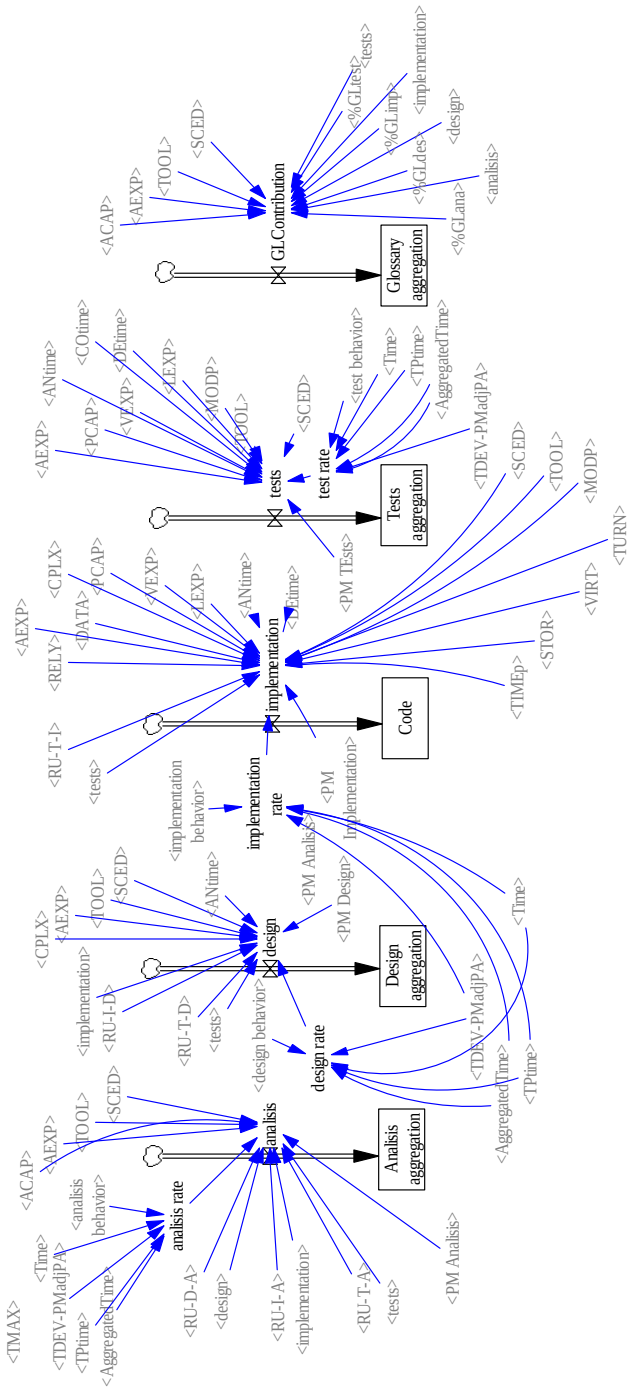

Figura 18 – Vista 2: Sub-modelo de la fase de "Construcción" en un proceso de desarrollo.

La tercera vista se ha agregado para representar el consumo paulatino del esfuerzo correspondiente a cada una de las etapas de la fase de construcción de software

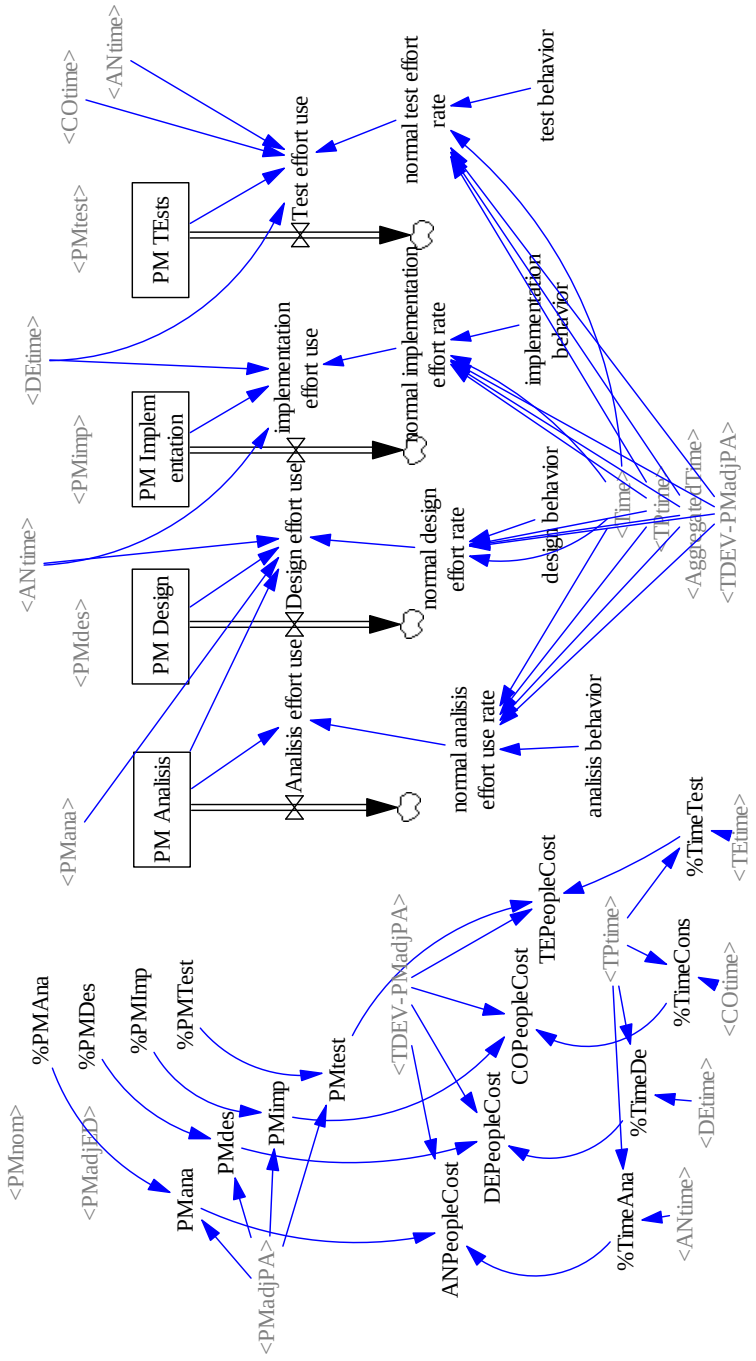

Figura 19 – Vista 3: Sub-modelo de distribución del esfuerzo en el proceso de desarrollo de software

3.6 Ejecución del modelo nominal

Se ejecutó el modelo con los parámetros "nominales", obteniéndose lo que se esperaría sea el comportamiento normal del sistema simulado[10].

Se pueden observar una gran cantidad de factores y su variación en el tiempo; sin embargo, enfocaremos 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 20, 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, aunque con ciertas simultaneidades, se concentran en su mayoría en distintas etapas 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 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*, ilustrado en la Figura 43, lo que nos da una primera idea de la aproximación de este modelo de simulación a los modelos teóricos existentes en la literatura.

[10] Como parte de las mejoras propuestas a este trabajo, se propone continuar con la experimentación, con el fin de lograr configurar las ecuaciones implícitas del modelo con una base empírica que conduzca a una mayor precisión.

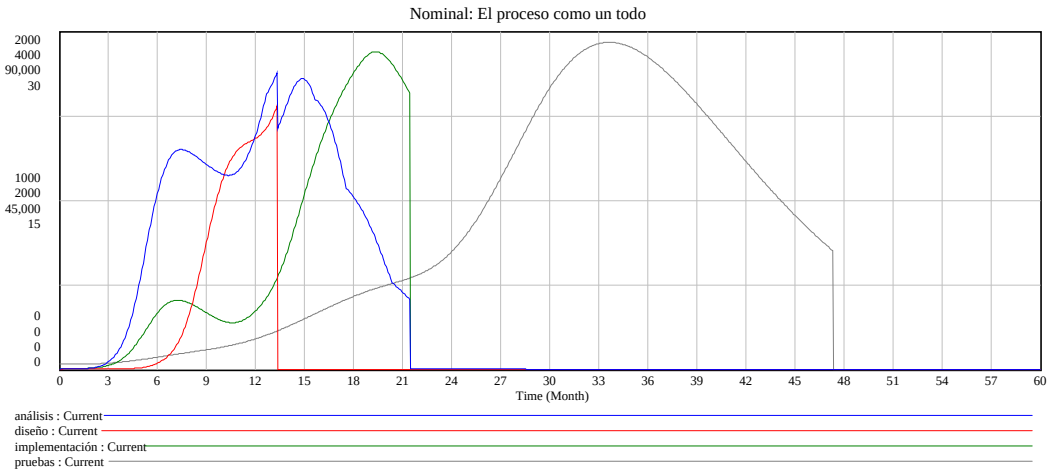

Figura 20 - Las fases de la "construcción" del proceso software: modelo nominal.

Si se piensa en las actividades del proceso de construcción como metas que deben ser alcanzadas en el transcurso del tiempo, la Figura 21, 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.

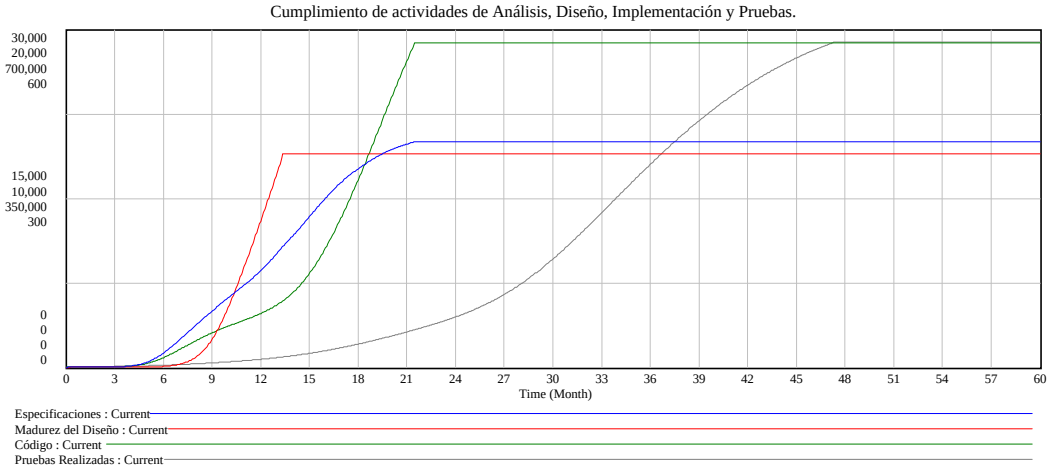

Figura 21 - Maduración de las fases del proceso de construcción: modelo nominal.

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 22, (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.

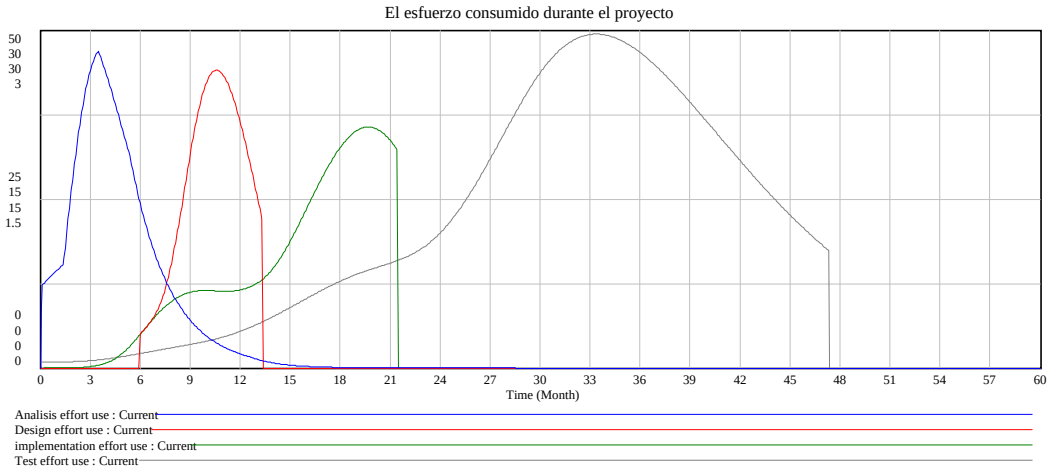

Figura 22 - Esfuerzo consumido en el proceso de construcción, separado por fases: modelo nominal.

4 Validación del modelo y estudio de escenarios

En este capítulo del presente trabajo, la intención es tomar el modelo de simulación diseñado, que se presentó en el capítulo anterior, y hacer una serie de pruebas que permitan verificar que los comportamientos e información que se puede apreciar a partir de la ejecución del modelo, se corresponde o tiene lógica, si se contrasta con el sistema real.

Dado que nuestro sistema real es un proceso de desarrollo de software, el modelo "nominal", cuyo comportamiento se presentó en la sección 1.1 ha sido calibrado con información generada por medio de encuestas realizadas a profesionales y académicos con un nivel alto de familiarización con el proceso de desarrollo de software, bien sea por medio de experiencias profesionales o por medio de investigación y experiencias académicas, pero siempre con experiencia mayor a 5 años. En el Anexo B, se encuentra el modelo de la encuesta que se realizó, y en el Anexo C se encuentran resumidos los resultados de la encuesta.

En la sección 4.1 se plantearán algunos escenarios que permiten hacer una primera validación del modelo de simulación aquí presentado, en base a la variación de parámetros de entrada del modelo.

En la sección 4.4 se plantearán varias hipótesis sobre los comportamientos esperados del modelo ante diferentes parámetros y/o configuraciones de los atributos. Seguidamente se harán las ejecuciones del modelo con estos atributos, para luego contrastar los resultados del modelo con las hipótesis planteadas y así obtener una estimación de la precisión y la robustez del modelo.

4.1 Estudio de escenarios: Primera aproximación

A continuación se van a hacer varias ejecuciones del modelo de simulación, cada una con una configuración específica de parámetros que corresponden a un escenario

específico. Se hará un particular esfuerzo en simular escenarios correspondientes a equipos de desarrollo posibles en el mundo de las organizaciones y bajo estudio en diversos estudios organizativos. Es así que se puede hablar de equipos colaborativos, cooperativos, adaptables, conflictivos y/o viciados.

Específicamente, se ejecutará el modelo en los siguientes escenarios:

4.1.1 Escenario 1: Equipo eficaz.

Un equipo eficaz de gestión desarrollo de software suele tener ciertas características que no sólo le describen sino que inciden directamente en el desempeño y resultados obtenidos de este equipo. Al ser eficaces, se caracterizan principalmente por un consumo preciso de recursos, alcanzando los objetivos en tiempos ajustados, y con una calidad aceptable y aproximada a la esperada. Los equipos eficaces, se limitan a alcanzar los objetivos planteados sin mayores esfuerzos por superar lo que esperan de ellos ni por optimizar el consumo de recursos.

4.1.2 Escenario 2: Equipo pasivo y viciado.

Un equipo pasivo y viciado, aunque logra los objetivos de desarrollo, suele tomar tiempos mayores a los estimados y con excesos en el consumo de recursos. Alcanza sus objetivos aunque no precisamente superando los estándares de calidad ni superando las expectativas. Los cambios en los objetivos o condiciones ambientales les afectan considerablemente y redundan en extensiones en los tiempos de gestión. Estos equipos suelen hacer poca reutilización de sus activos y al depender de desarrollos completos en casi su totalidad requieren de mayor trabajo, dedicación y recursos para lograr por lo menos eficacia en la consecución de objetivos.

4.1.3 Escenario 3: Equipo colaborativo y eficiente.

Un equipo colaborativo y adaptable, además de ser eficiente, alcanza los objetivos de desarrollo planteados, supera las expectativas, reduce considerablemente los tiempos consumidos para obtener los productos y subproductos, reutiliza los activos disponibles en sus bibliotecas y mantiene actitudes resolutivas, cooperativas y con disposición para adaptarse a los cambios con el mínimo impacto.

	Descripción de escenario	Parámetros a variar
Escenario 1	Se trata de un escenario en que el equipo es mejor que el equipo promedio y se siente en un entorno familiar. Los programadores y analistas se sienten como en el entorno "orgánico" descrito en (California., 2003).	ACAP: 1→0.85 AEXP: 1→0.91 PCAP: 1→0.70 LEXP: 1→0.95 TOOL: 1→0.83 SCED: 1→1.04
Escenario 2	Se trata de un escenario en que el equipo es regular-malo y con restricciones. El personal cuenta con experimentación heterogénea y un entorno ni familiar ni extraño sino cuasi neutro. Los programadores y analistas se sienten como en el entorno "semi-libre" o "semi-encajado" descrito en (California., 2003).	ACAP: 1→1.19 AEXP: 1→1.13 PCAP: 1→1.17 LEXP: 1→1.14 TOOL: 1→1.10 SCED: 1→1.22
Escenario 3	Se trata de un escenario en que el equipo está altamente capacitado, tanto en cuanto a personal como a especificaciones técnicas. Existe cierto grado de experimentación y los programadores y analistas se sienten en un entorno "libre".	ACAP: 1→0.71 AEXP: 1→0.82 PCAP: 1→0.70 LEXP: 1→0.90 TOOL: 1→0.83 SCED: 1→1.10

Tabla 11 - Validación del modelo nominal - Variación de parámetros

4.1.4 Análisis del Escenario 1

Al variar algunos parámetros según se ha especificado en la Tabla 11, se obtienen variaciones en el comportamiento del modelo que permiten concluir sobre el efecto de los elementos que esos parámetros representan sobre el sistema real.

En la Figura 23, se observa que tras hacer unas ligeras variaciones de los parámetros referentes al modelo nominal, y configurando éstos de manera que representen un equipo medianamente mejor que la media (representada en la Figura 20), los tiempos de ejecución de las actividades se ven reducidos, así como las magnitudes de los flujos de información. En otras palabras, un equipo ligeramente mejor representará ahorros en tiempos y en los flujos de información del proceso.

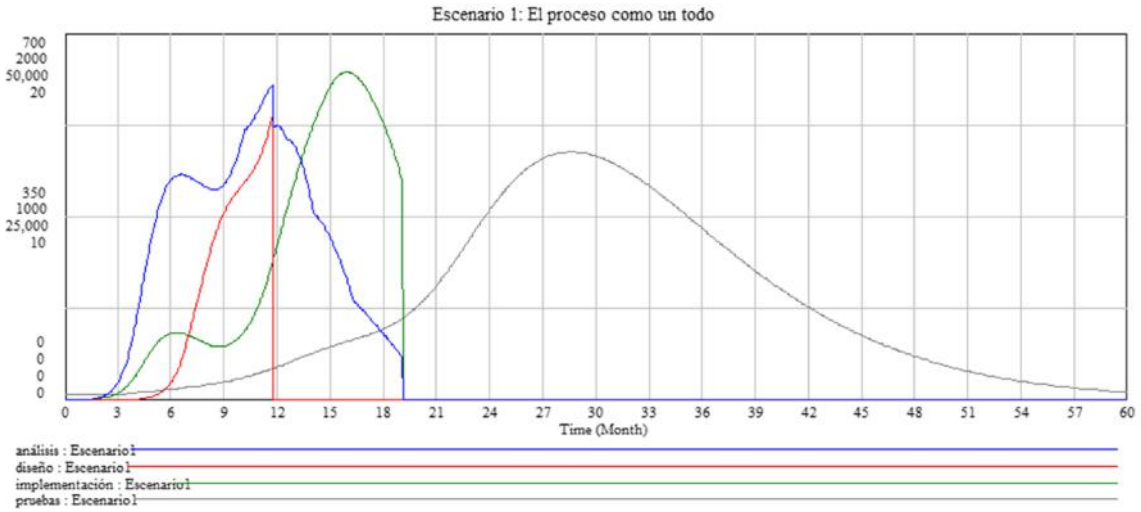

Figura 23 - Las fases de la "construcción" del proceso software: Escenario 1.

El cumplimiento de las actividades, en este primer escenario (Figura 24) es ligeramente más eficiente que en el caso nominal, lo que se puede observar tanto por la reducción en los tiempos que se toma el sistema en completar las fases (eje horizontal) como por la reducción de magnitudes de información manejadas (véase las escalas del eje vertical), si se compara con la Figura 21.

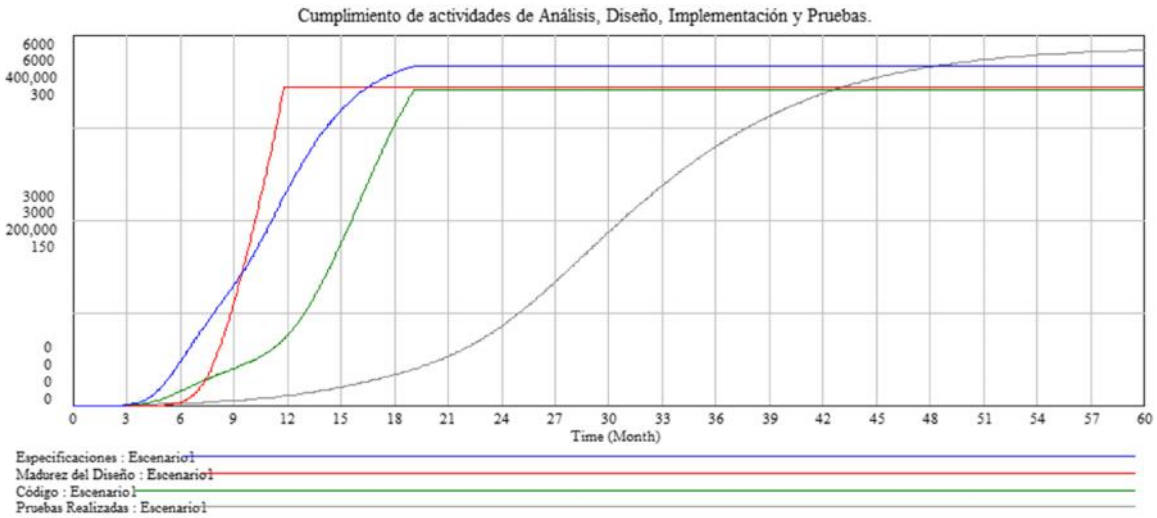

Figura 24- Maduración de las fases del proceso de construcción: Escenario 1.

El esfuerzo en este primer escenario también es significativamente mejor usado que en el caso nominal. En la Figura 25, vemos que en menor tiempo se dedicó el esfuerzo a las

actividades y con consumos ligeramente menores, como lo indican las escalas del eje vertical, si comparado con la Figura 22.

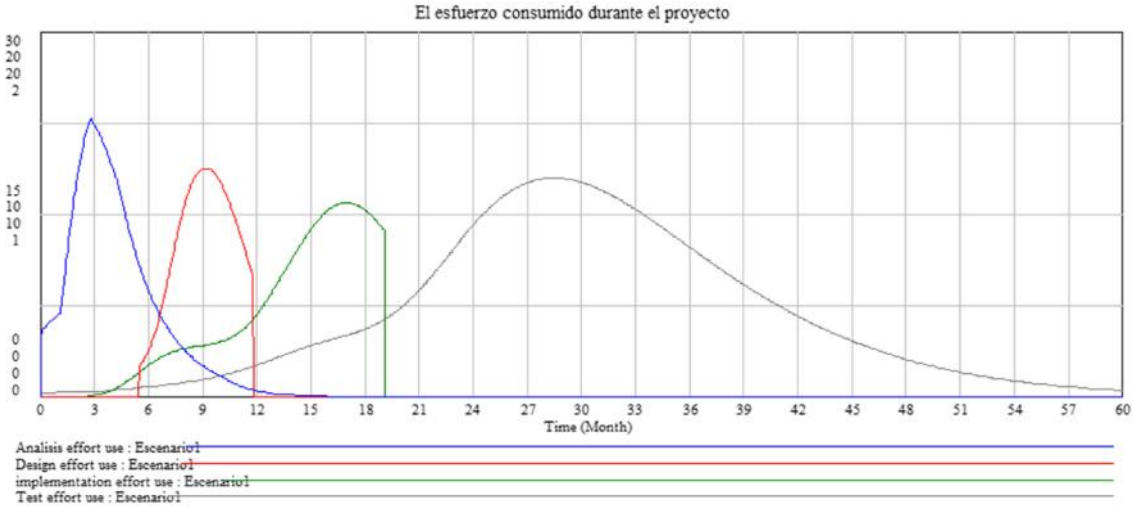

Figura 25 - Esfuerzo consumido en el proceso de construcción, separado por fases: Escenario 1.

En la Figura 26, se puede observar como el sistema, a partir de la estimación del esfuerzo realizado con las ecuaciones del modelo COCOMO, parte de cierta cantidad de esfuerzo para cada una de las fases de la construcción. Si se recorre el eje horizontal (tiempo) mirando hacia las curvas del esfuerzo disponible, vemos que, como se esperaría, el primer proceso en comenzar a consumir esfuerzo es el análisis, seguido consecutivamente por diseño, implementación y finalmente pruebas (siempre en distintas escalas, dado que unos subprocesos requieren de mayor esfuerzo que otros).

Figura 26 - Des-acumulación del "Esfuerzo" como recurso del proceso software, por fases. Escenario 1.

4.1.5 Análisis del Escenario 2

Este segundo escenario, cuyo funcionamiento se muestra en la Figura 27, representativo de un equipo menos eficiente, con valores de parámetros de COCOMO que se corresponden con peores parámetros de equipo (ACAP: 1 →1.19, AEXP: 1→1.13, PCAP: 1→1.17, LEXP: 1→1.14, TOOL: 1→1.10, SCED: 1→1.22), produce dilataciones en los tiempos de ejecución de las fases del proceso (están más extendidos en el eje horizontal) e incrementos importantes en los flujos de información (están más extendidos en el eje vertical), cuando es comparado con el modelo nominal representado en la Figura 20, y con el modelo ligeramente mejor que el nominal (escenario 1), representado en la Figura 23.

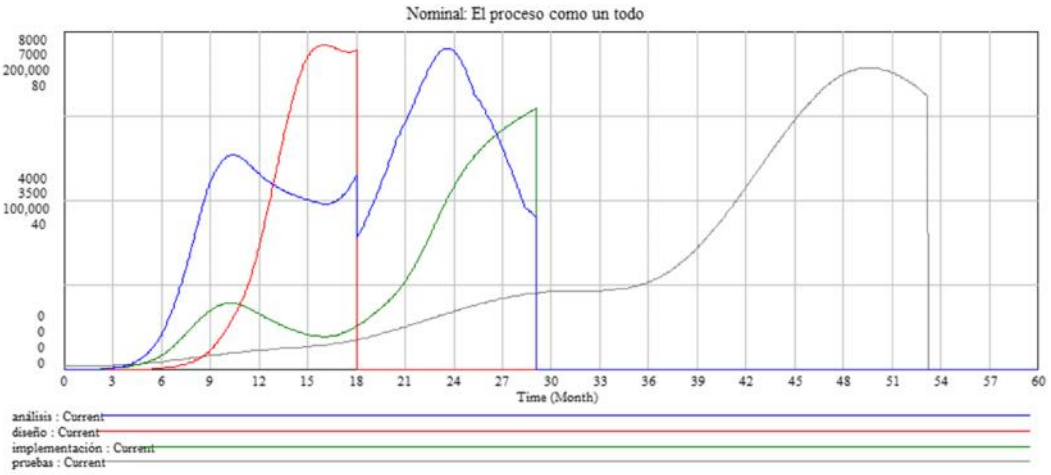

Figura 27- Las fases de la "construcción" del proceso software: Escenario 2.

Como es de esperarse, al observar el comportamiento de los niveles correspondientes a la maduración de las fases del proceso de construcción de software, la principal diferencia con los casos nominal (Figura 21) y escenario 1 (Figura 24), es que aquí, el empeoramiento de los parámetros causa que el proceso "madure" más lentamente, como se aprecia en la Figura 28.

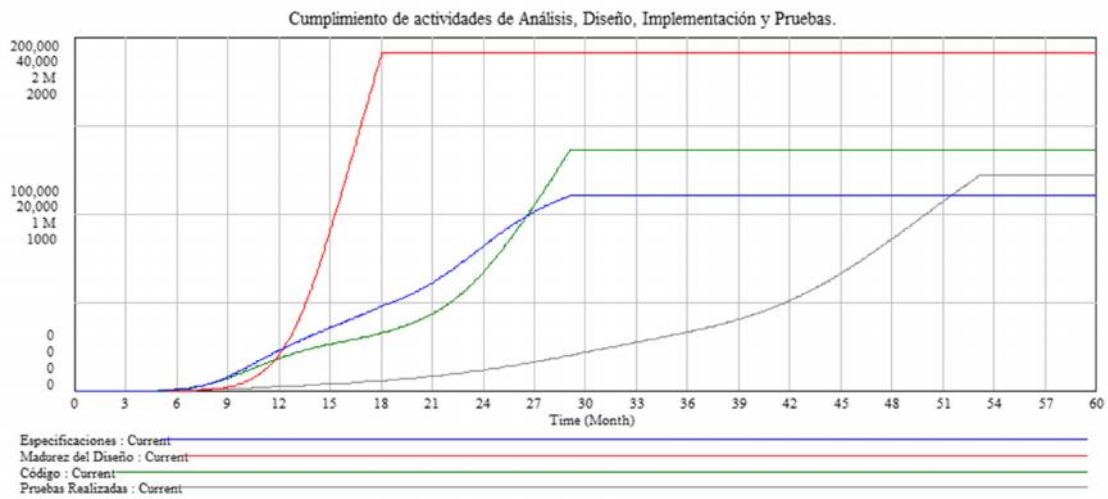

Figura 28- Maduración de las fases del proceso de construcción: Escenario 2.

De manera similar a los dos casos anteriores, en este segundo escenario, se ve que el esfuerzo distribuido se corresponde con la ocurrencia de actividades según la fase del proceso de construcción que se desarrolle en el transcurso del tiempo. En la Figura 29 se aprecia que la distribución del esfuerzo se dilata en el tiempo, como es de esperarse cuando se trata de un equipo menos eficiente.

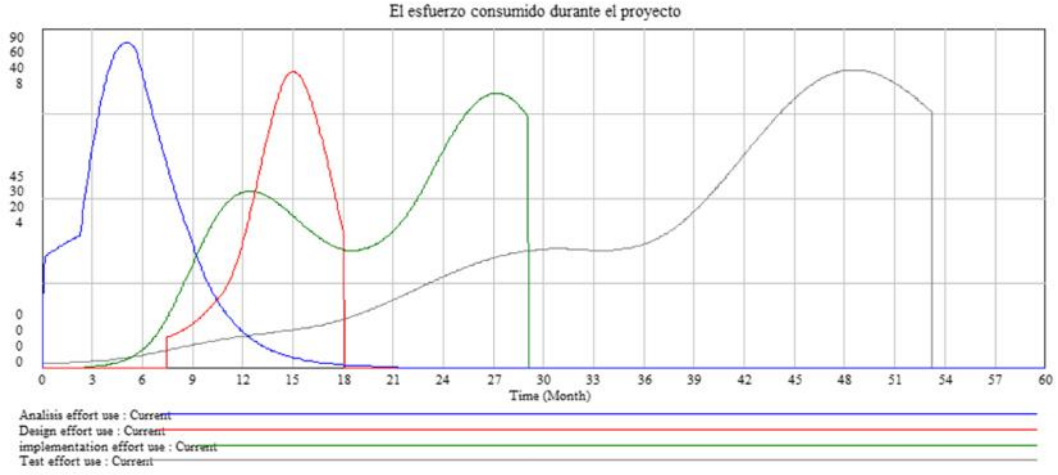

Figura 29 - Esfuerzo consumido en el proceso de construcción, separado por fases: Escenario 2.

4.1.6 Análisis del Escenario 3

Ahora se presentará el mejor de los casos posibles, con la variación de parámetros propuesta. Este tercer escenario se corresponde con el de un equipo de desarrollo eficiente, efectivo, con buen manejo de herramientas y buen nivel de organización y

programación de trabajo interno, cuyos parámetros de COCOMO serían: ACAP: 1→0.71, AEXP: 1→0.82, PCAP: 1→0.70, LEXP: 1→0.90, TOOL: 1→0.83, SCED: 1→1.10.

En la Figura 30, se presenta el proceso de construcción como un "todo". Si lo comparamos con las gráficas Figura 20, Figura 23 y Figura 27, podemos ver que es el que presenta menores tiempos de ejecución y con un menor flujo de información para cada grupo de actividades.

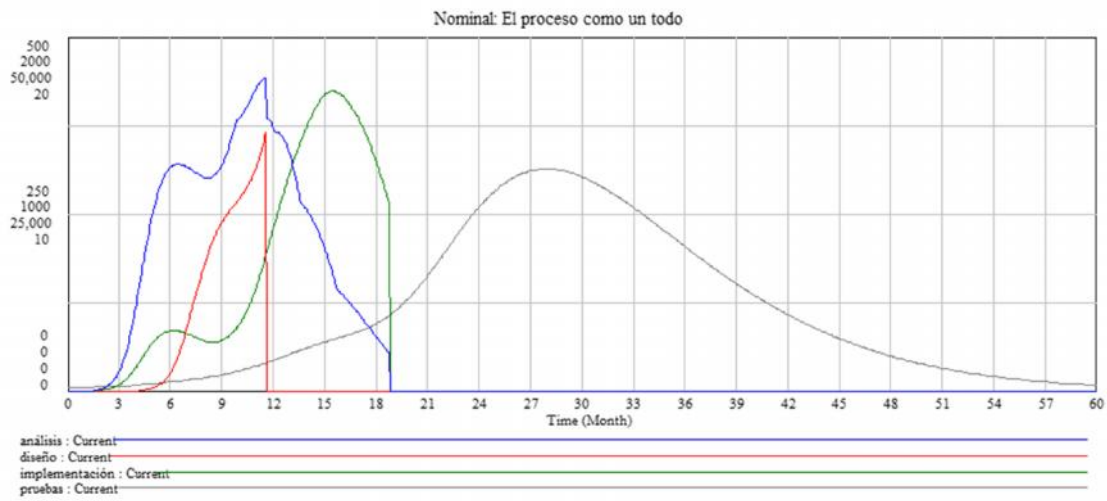

Figura 30 - - Las fases de la "construcción" del proceso software: Escenario 3.

Siendo el mejor de los casos (este escenario 3), la Figura 31 permite corroborar que la maduración de los sub-procesos se lleva a cabo de manera más rápida y con menos flujo de información, demostrando que efectivamente, un mejor equipo de trabajo será más eficiente, como lo indica el modelo; esto a partir de comparaciones con las imágenes de Figura 21, Figura 24, y Figura 28.

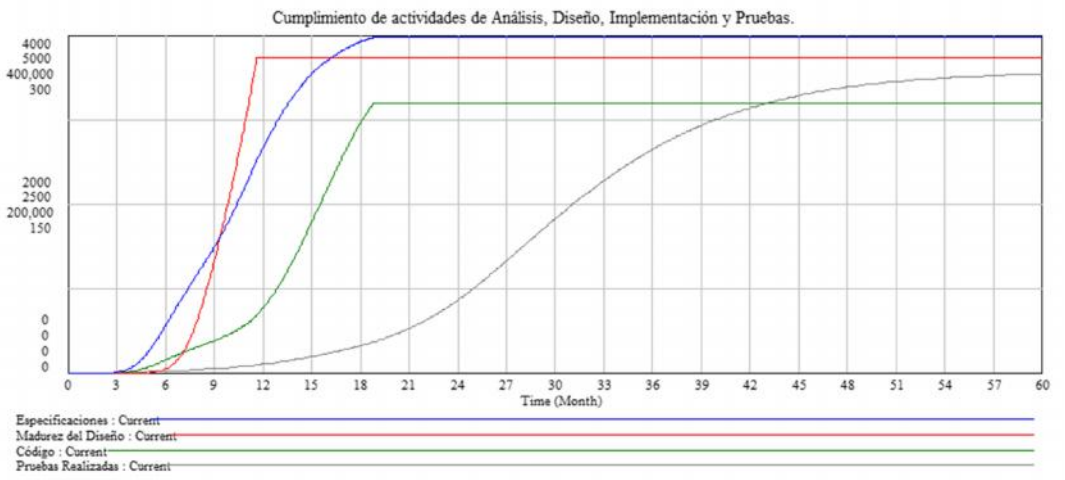

Figura 31 - - Maduración de las fases del proceso de construcción: Escenario 3.

La Figura 32 permite observar que el esfuerzo distribuido se usa según correspondería al proceso teórico, en que el orden de concentración del esfuerzo es: en el análisis, en el diseño, en la implementación, y finalmente en las pruebas. Sin embargo, el aporte de esta imagen está en que permite apreciar que con menores esfuerzos (eje vertical) los sub-procesos se alcanzaron a desarrollar de manera más rápida (eje horizontal) que en los casos anteriores (representados en Figura 22, Figura 25 y Figura 29).

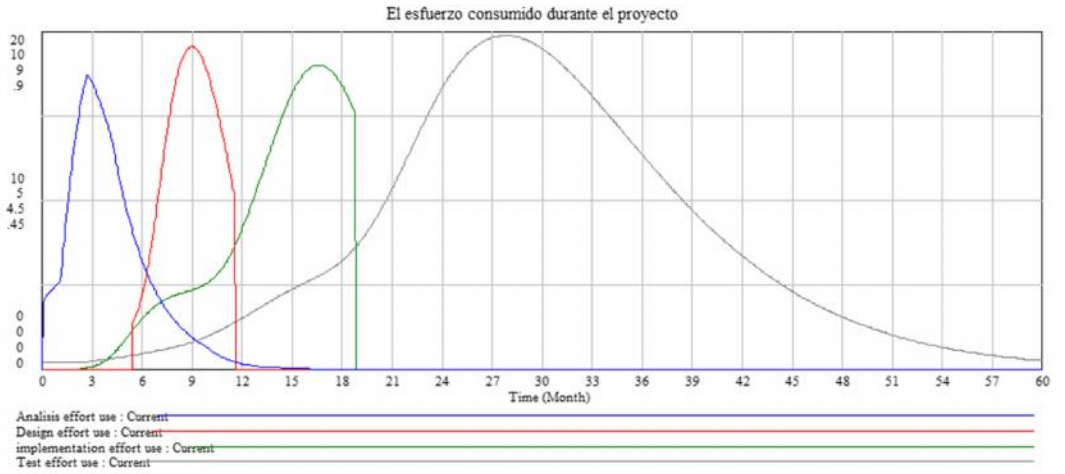

Figura 32- Esfuerzo consumido en el proceso de construcción, separado por fases: Escenario 3.

En este tercer escenario se quiere resaltar cómo el uso del esfuerzo, como recurso, durante la vida del proceso, se enfoca inicialmente en el análisis. Cuando ya se ha dedicado suficiente al análisis, se comienza con el diseño y ligeramente después con la

implementación. Finalmente, luego de que existe un prototipo de producto, comienza a usarse el esfuerzo en las pruebas, terminando de consumir el recurso "esfuerzo" disponible.

Figura 33- Des-acumulación del "Esfuerzo" como recurso del proceso software, por fases. Escenario 3.

4.2 Costes del Proyecto

Basados en la estimación de costes de COCOMO, es posible estudiar cómo se ven afectados los recursos requeridos para llevar a cabo el proyecto. A continuación se presenta un análisis de la variación de los costes de proyecto con respecto a los escenarios 1, 2 y 3 de la sección 4.1.

Para las configuraciones de parámetros que se muestra en Tabla 11 y Tabla 12, la variación de los costes, en términos de personas requeridas para llevar a cabo el proyecto es la que se muestra en la Figura 34 - Variación de costes en función de los escenarios 1, 2 y 3. Unidades: Personas..

Tabla 12 - Variación de parámetros en el estudio de costes.

Escenario 1	**Escenario 2**	**Escenario 3**
Equipo eficaz	**Equipo pasivo**	**Equipo eficiente**
ACAP: 1→ 0.85	ACAP: 1→ 1.19	ACAP: 1→ 0.71
AEXP: 1→ 0.91	AEXP: 1→ 1.13	AEXP: 1→ 0.82
PCAP: 1→ 0.70	PCAP: 1→ 1.17	PCAP: 1→ 0.70
LEXP: 1→ 0.95	LEXP: 1→ 1.14	LEXP: 1→ 0.90
TOOL: 1→ 0.83	TOOL: 1→ 1.10	TOOL: 1→ 0.83
SCED: 1→ 1.04	SCED: 1→ 1.22	SCED: 1 →1.10

Para el caso nominal del modelo calibrado, la estimación del proyecto indica que se requieren alrededor de 20 personas (línea azul) para la ejecución total del proyecto. Tras configurar el modelo para distintos escenarios (Véase las variaciones de los parámetros en la Tabla 12), para el caso del equipo eficaz (escenario 1: ligeramente mejor que el nominal), disminuye el coste a aproximadamente 11 personas, mientras que para el caso del equipo pasivo y menos preparado (escenario 2) el coste se incrementa a aproximadamente 30 personas. Para el mejor de los casos, es decir, en el que se presenta el mejor equipo y personal (escenario 3), el requerimiento de personas para el proyecto disminuye a aproximadamente 9.

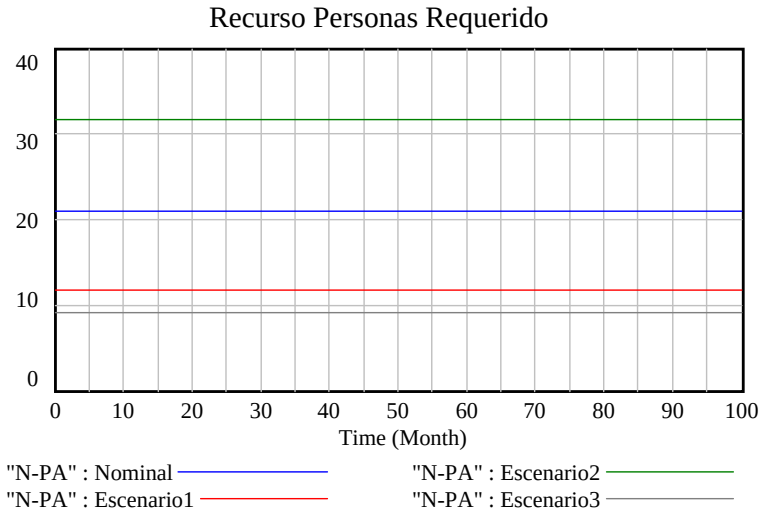

Figura 34 - Variación de costes en función de los escenarios 1, 2 y 3. Unidades: Personas.

Si resultase de interés conocer la estimación de costes en términos monetarios, en este modelo se implementó un multiplicador configurable que simplemente transforma el coste en personas a costes en moneda por medio de una variable auxiliar. Si se establece que el coste por persona es de 4500 unidades monetarias, la gráfica de costes en estos términos es análoga a la anterior, sólo que transformada a la escala monetaria de interés, como en la Figura 35. En este caso, se ve que para el caso en que se tiene mejores analistas, experiencia y equipos, el coste del proyecto es significativamente menor (< 50000 unidades monetarias) que el caso los otros tres casos; de igual manera, para el peor de los casos, el coste es significativamente mayor que para los otros tres casos (cerca 150000 unidades monetarias).

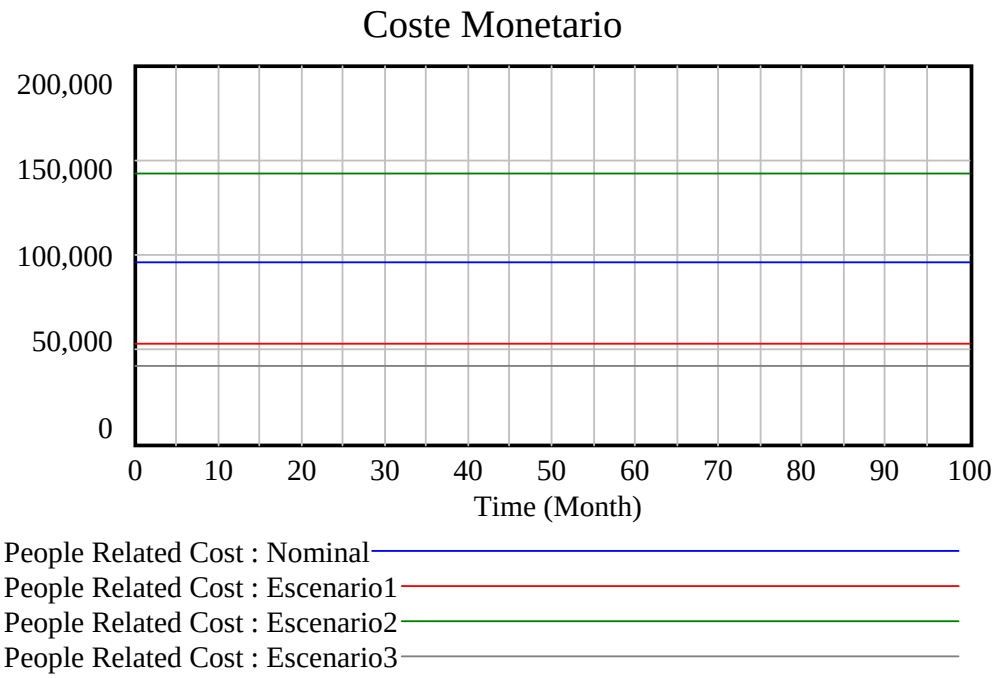

Figura 35 Costes transformados a términos monetarios.

4.3 Otras hipótesis de validación

En correspondencia con la realidad de los equipos de desarrollo de software, y en muchos aspectos con equipos de trabajo en general, el modelo creado debería ser capaz de mostrar comportamientos variantes en el tiempo de acuerdo a la calibración de

ciertos atributos o aspectos que, al igual que en el sistema real, inciden en el funcionamiento general del sistema. Ejemplos de aspectos que tienen este tipo de incidencia en el funcionamiento del sistema son: Complejidad del producto software a desarrollar, Capacidad de análisis y programación del equipo, familiarización con el lenguaje y máquinas virtuales que se usan en el proceso de desarrollo, etc.

A continuación se presenta una serie de hipótesis, sobre el funcionamiento del sistema, bajo ciertas condiciones específicas que se pretende argumentar o refutar a partir del contraste entre el modelo de simulación y los comportamientos que muestra en su versión nominal y el mismo modelo ejecutado con la variación de ciertos parámetros.

Nº de Hipótesis	**Parámetros a variar**	**Consecuencia esperada (declaración)**
01	Todos sobre personal	Cuanto mejor es la preparación de todo el equipo, menor será el esfuerzo requerido para llevar a cabo el proyecto.
02	Todos sobre Equipamiento tecnológico	Cuanto peor es el manejo y capacidades sobre el equipamiento tecnológico que se usa para desarrollar el producto, mayor será el esfuerzo requerido.
03	Complejidad del producto	Cuanto mayor es la complejidad de un producto software, mayor el tiempo de análisis y diseño ocupado en él.
04	Complejidad del producto	Cuanto mayor es la complejidad de un producto software, mayor el tiempo en comenzar la fase de realización de pruebas.
05	Complejidad del producto	Cuanto mayor es la complejidad del producto, mayor será la cantidad de términos que se incorporen al glosario construido.

Tabla 13 - Hipótesis de validación del modelo

4.4 Uso de hipótesis de validación.

A continuación se presenta el resultado de varios experimentos sobre el modelo de simulación que permiten concluir sobre la validez de éste con respecto a las hipótesis planteadas. Veamos cada uno de estos contrastes y las conclusiones sobre el modelo.

4.4.1 Nominal Vs. Hipótesis 1: Variación de parámetros de personal, efecto en todo el proceso.

Para llevar a cabo esta prueba se re-calibra y ejecuta el modelo con los siguientes valores de parámetros específicos que se corresponden con una mejor preparación y capacidades del equipo.

Parámetro a variar	Valor nominal	Valor experimental
Capacidad de análisis	1.00	0.86
Experiencia en la aplicación	1.00	0.91
Calidad de los programadores	1.00	0.86
Experiencia en la máquina virtual	1.00	0.90
Experiencia en el lenguaje	1.00	0.95

Tabla 14 - Valores de parámetros para prueba de hipótesis de validación 1

Tras ejecutar el modelo, es de interés observar el comportamiento del sistema, con la intención de observar cómo se ve modificado el sistema ante los cambios de parámetros efectuados. En la Figura 36 se puede ver cómo, para las cuatro etapas del proceso, el principal efecto de los cambios realizados es la reducción en los flujos de información y en el tiempo que toman las actividades en ser realizadas.

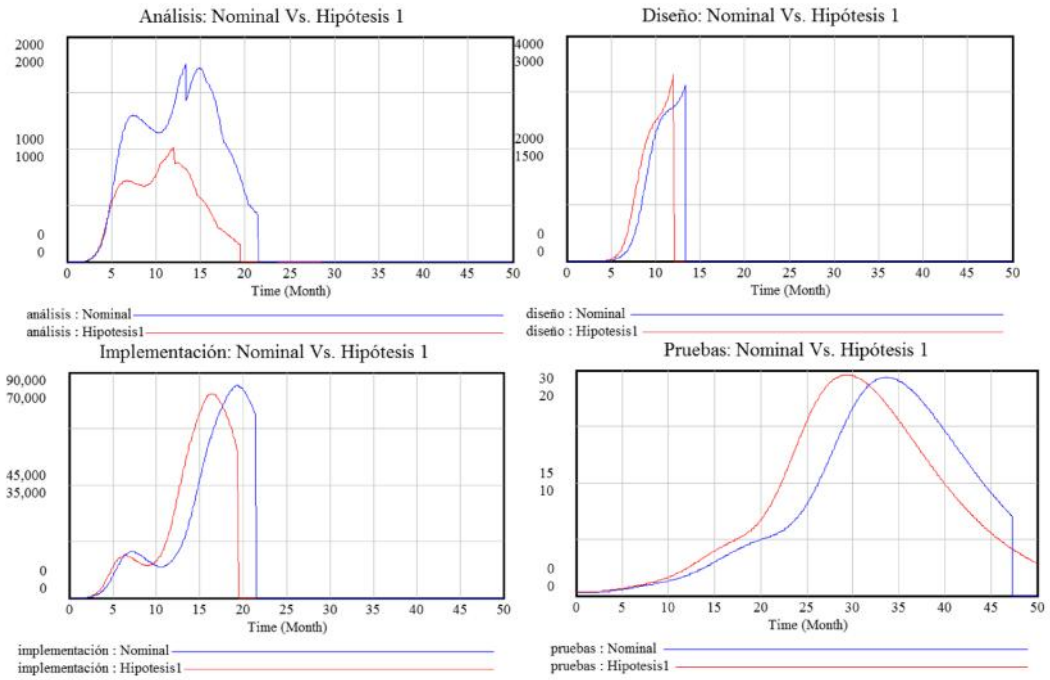

Figura 36 - Las cuatro etapas de la fase "construcción". Modelo Nominal Vs. Hipótesis 1

De manera muy similar, si lo que se observa es el esfuerzo que se emplea en cada una de las etapas, como se presenta en la Figura 37, es posible ver dos cambios

fundamentales: el primer cambio se refiere a la reducción generalizada de los tiempos (eje horizontal) de consumo del esfuerzo, y el segundo cambio es referente al flujo de esfuerzo propiamente dicho, ya que se ve significativamente reducido (eje vertical) tras la modificación de los parámetros.

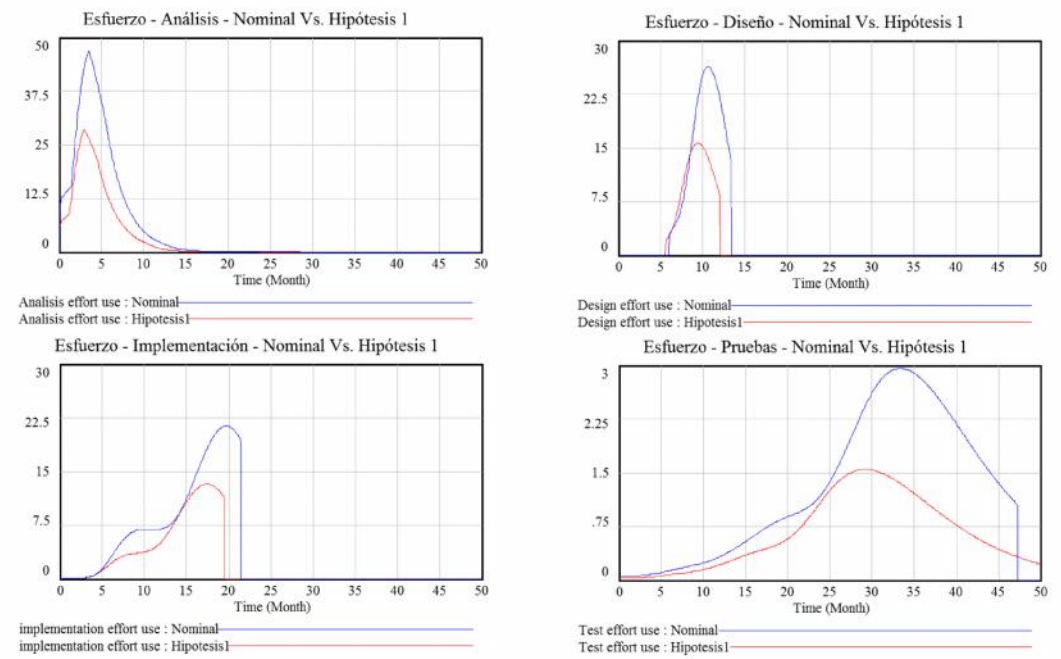

Figura 37- Esfuerzo distribuido - Fase "construcción". Modelo Nominal Vs. Hipótesis 1

Como se ha podido ver en los gráficos de Figura 36 y Figura 37, para este caso en que se han cambiado los parámetros *Capacidad de análisis*, *Experiencia en la aplicación*, *Calidad de los programadores*, *Experiencia en la máquina virtual* y *Experiencia en el lenguaje*, el resultado es una mejoría en el consumo de los recursos, como sería de esperar para un equipo con estos cambios en los parámetros, confirmando la hipótesis 1 aquí planteada.

4.4.2 Nominal Vs. Hipótesis 2: Variación de parámetros de equipamiento: efecto en todo el proceso.

Para llevar a cabo esta prueba se re-calibra y ejecuta el modelo con los siguientes valores de parámetros específicos que se corresponden con un peor equipamiento y capacidades de los equipos informáticos.

Parámetro a variar	Valor nominal	Valor experimental
Restricciones de tiempo de ejecución	1.00	1.30
Restricciones de memoria virtual	1.00	1.21
Volatilidad de la máquina virtual	1.00	1.30
Tiempo de respuesta	1.00	1.15

Tabla 15 - Valores de parámetros para prueba de hipótesis de validación 2

En la Figura 38 se observa el comportamiento en el transcurso del tiempo de las cuatro etapas de la fase construcción, bajo las dos configuraciones de parámetros señaladas. En las cuatro gráficas que se muestran se puede observar que el efecto de variar los parámetros correspondientes al equipamiento y capacidades informáticas. En primera instancia, se puede apreciar que existe una diferencia importante en los tiempos en que se inician, desarrollan y terminan las etapas del proceso; y en segunda instancia, hay una diferencia importante en cuanto la magnitud de los flujos de información, siendo menor esta magnitud para el caso en que se cuenta con mejor equipamiento informático.

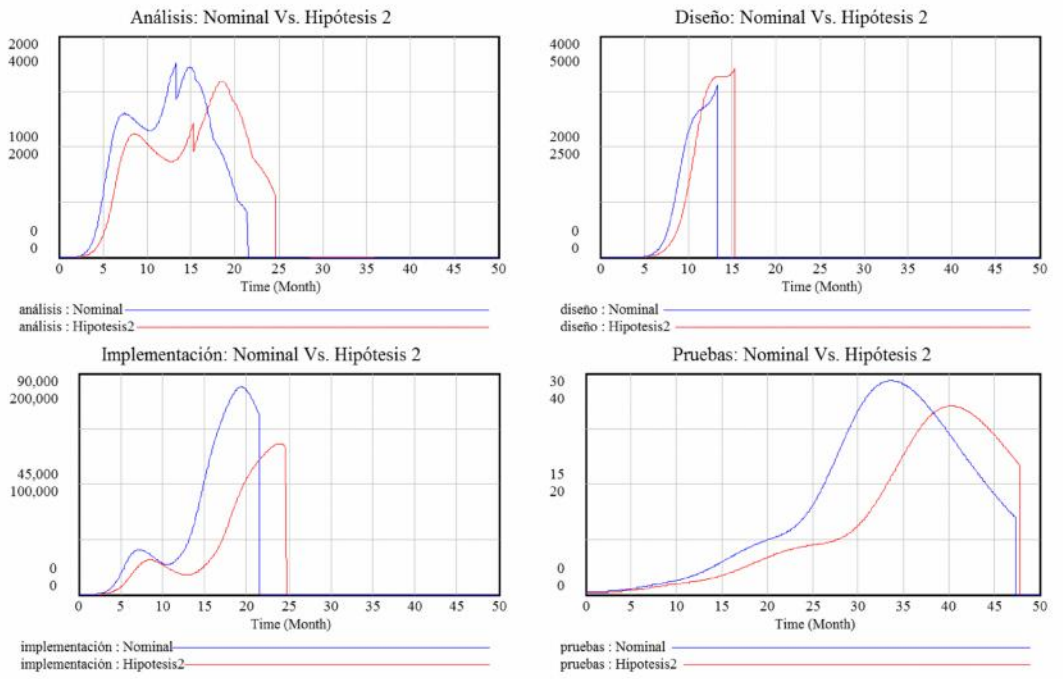

Figura 38- Las cuatro etapas de la fase "construcción". Modelo Nominal Vs. Hipótesis 2

De importancia para comprender el efecto de la variación planteada, es observar y analizar cómo es el esfuerzo dedicado a las distintas etapas del proceso, lo que podemos apreciar en la Figura 39. Como es de esperarse, en las cuatro gráficas el modelo de

simulación sugiere que cuanto mejor es el equipamiento informático, menor será el esfuerzo requerido para ejecutar estas fases, lo que se hace evidente al observar las diferencias de los valores en el eje vertical.

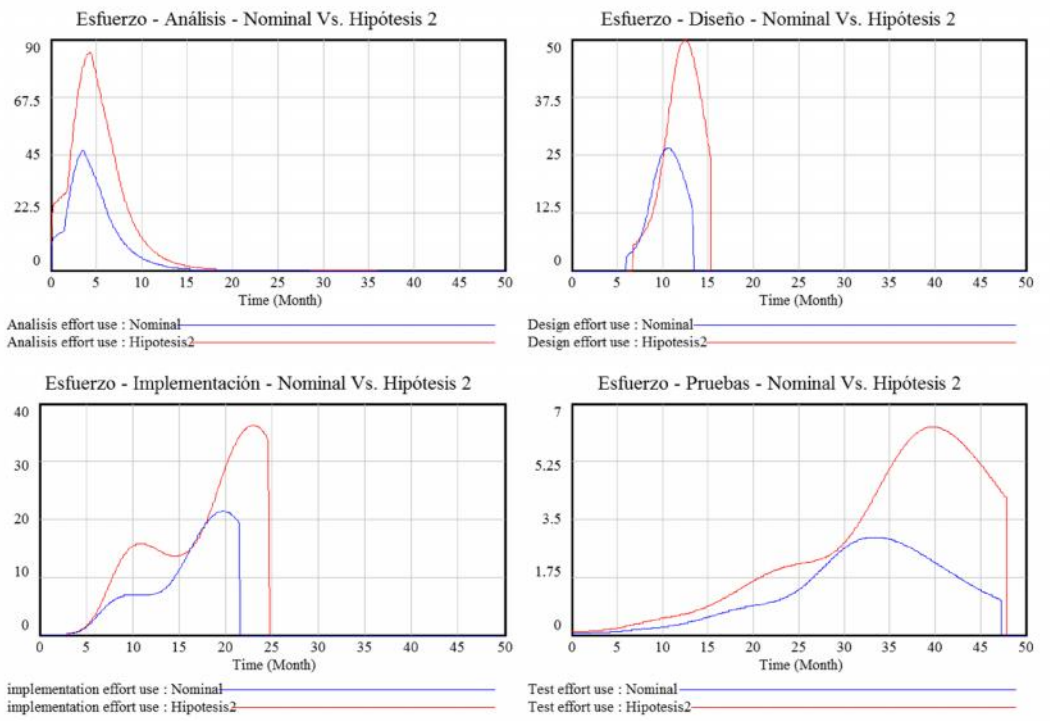

Figura 39 - Esfuerzo distribuido - Fase "construcción". Modelo Nominal Vs. Hipótesis 2

4.4.3 Nominal Vs. Hipótesis 3: Variación de la complejidad y efecto en el "análisis" y "diseño".

Para llevar a cabo esta prueba se re-calibra y ejecuta el modelo con el siguiente valor de parámetro específico que se corresponden con una mayor complejidad del proyecto. Aquí observamos cómo se ve afectado el tiempo dedicado al análisis y diseño en el proceso software.

Parámetro a variar	Valor nominal	Valor experimental
Complejidad general	1.00	1.30

Tabla 16 - Valores de parámetros para prueba de hipótesis de validación 3

La Figura 40 presenta los comportamientos de las cuatro etapas de la fase construcción cuando se modifica el parámetro referente a la complejidad. Como se puede apreciar, el parámetro "complejidad" produce el efecto de dilatar tanto los tiempos de ejecución de

cada tipo de actividades, como los flujos de información requeridos para llevar a cabo las etapas.

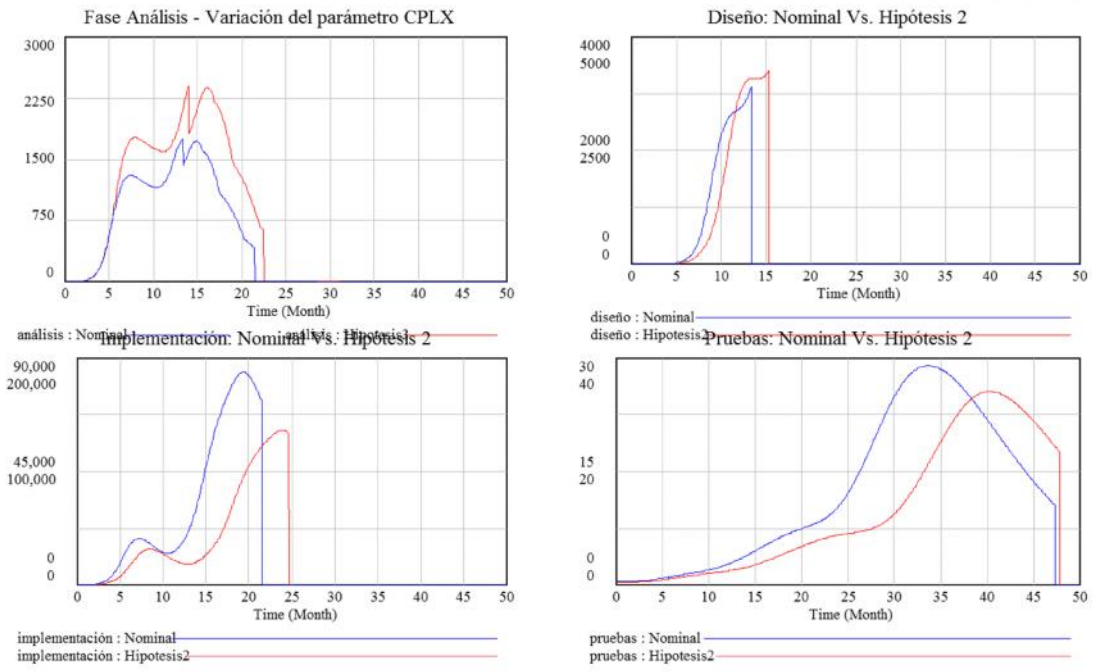

Figura 40 - Efecto de la variación de la Complejidad (CPLX) en las etapas de "construcción" del proceso software.

4.4.4 Nominal Vs. Hipótesis 4: Variación de la complejidad y efecto en la etapa de "pruebas".

Para llevar a cabo esta prueba se re-calibra y ejecuta el modelo con el siguiente valor de parámetro específico que se corresponden con una mayor complejidad del proyecto. En este caso, a diferencia de la hipótesis anterior en que observábamos el tiempo para las fases de análisis y diseño fundamentalmente, observaremos con énfasis la etapa de las pruebas y el tiempo que ocurre desde que comienza la simulación hasta que se activan las pruebas.

Parámetro a variar	Valor nominal	Valor experimental
Complejidad general	1.00	1.30

Tabla 17 - Valores de parámetros para prueba de hipótesis de validación 4

En la Figura 41 se aprecia que el aumento de la complejidad propuesto ocasiona principalmente que se disminuya el flujo de pruebas que se hacen durante todo el proceso. La ocurrencia en el tiempo de la etapa de realización de las pruebas es

ligeramente más rápida en el caso menos complejo, sin embargo no sería una diferencia muy significativa.

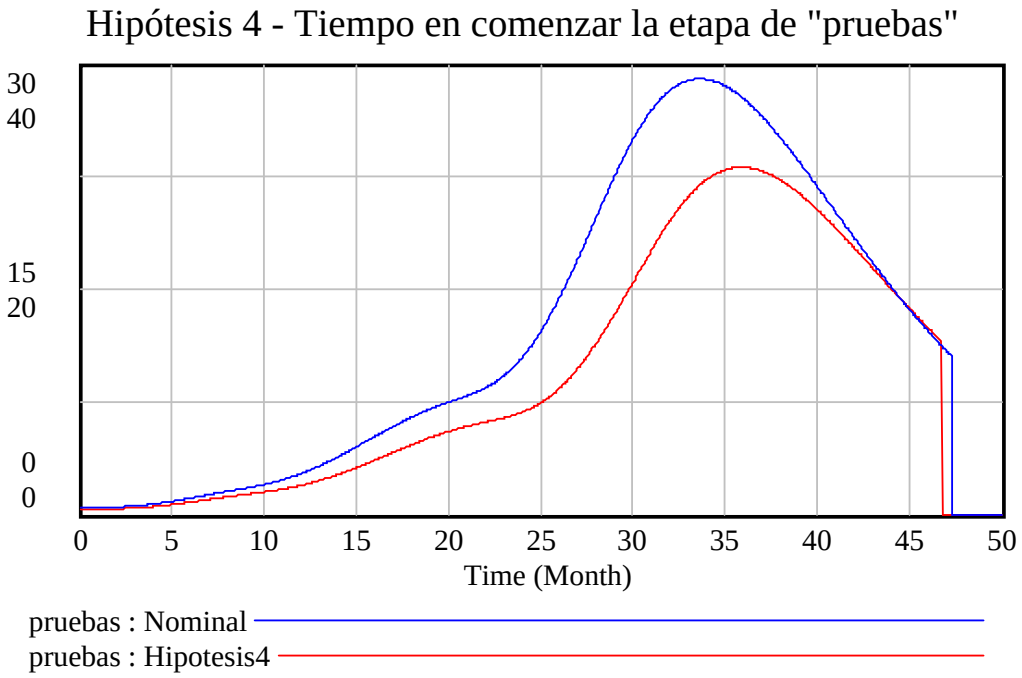

Figura 41 - Hipótesis 4: Variación del tiempo en comenzar a ejecutarse la etapa de pruebas.

4.4.5 Nominal Vs. Hipótesis 5: Variación de la complejidad y efecto en la creación del glosario.

Para llevar a cabo esta prueba se re-calibra y ejecuta el modelo con el siguiente valor de parámetro específico que se corresponden con una mayor complejidad del proyecto. Aquí observamos cómo se ve afectado el glosario que se construye y su relación directamente proporcional con la complejidad del proyecto.

Parámetro a variar	Valor nominal	Valor experimental
Complejidad general	1.00	1.30

Tabla 18 - Valores de parámetros para prueba de hipótesis de validación 6

En la Figura 42 se puede apreciar tanto el flujo de aporte al glosario como el crecimiento del glosario en sí mismo. En el caso más complejo (línea roja) existe un flujo ligeramente mayor de aportes al glosario, y también el glosario es mucho mayor que el del caso nominal.

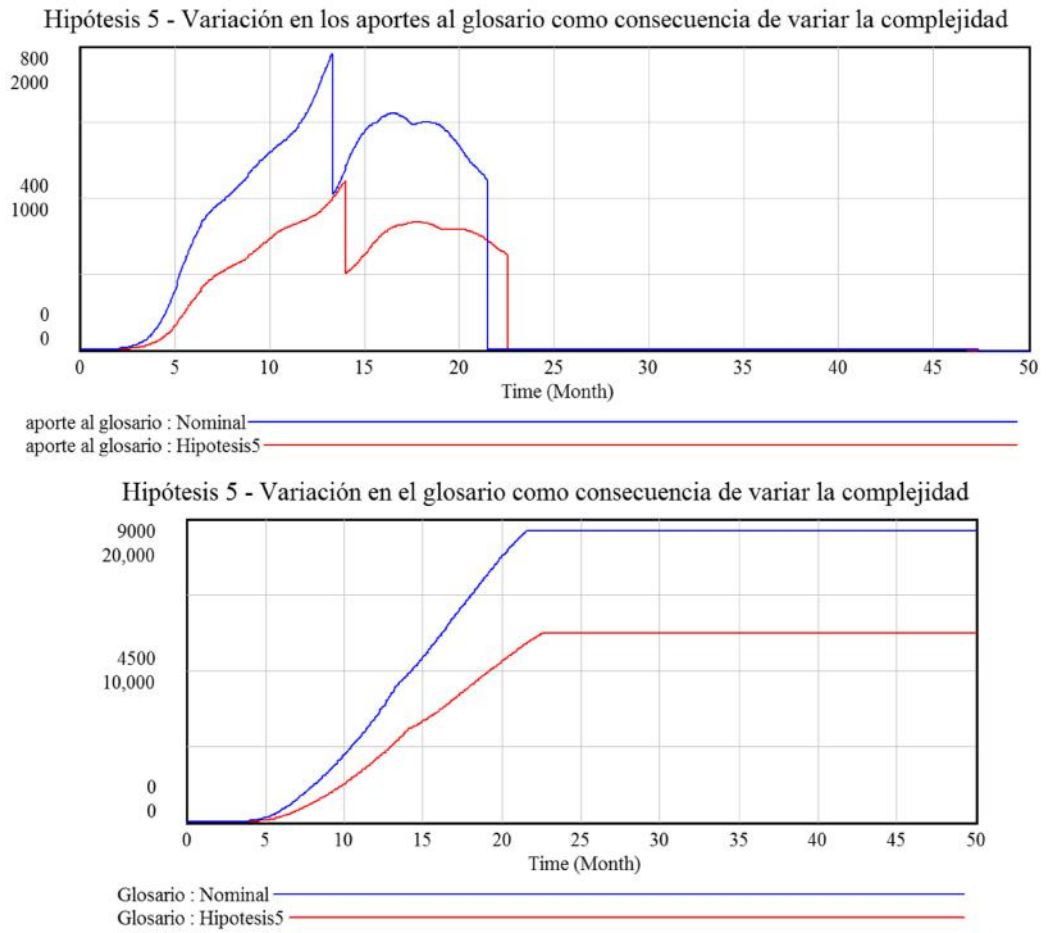

Figura 42 - Hipótesis 5: Variación del nivel Glosario y el flujo "Aporte al glosario" como consecuencia de variar la complejidad.

4.5 El proceso en general: Referencia Vs. Modelo

4.5.1 El proceso software. RUP: un modelo de referencia de su ejecución

A continuación veremos cómo es el comportamiento del modelo con respecto a un proceso general de desarrollo de software, tal como está descrito en toda la literatura pero a efectos de este trabajo revisado en (Pressman, 2010) y en (Spasic & Onggo, 2012).

El proceso de desarrollo de software en general, sigue un comportamiento con fases no tan exactamente asíncronas, aunque sí claramente diferenciadas en el tiempo y con un comportamiento variable dependiendo de cada proyecto en particular. Por lo general se puede tomar como referencia de la ejecución de las distintas fases en el transcurso del

tiempo, el clásico gráfico de ejecución del Rational Unified Process (RUP), del que el Método de Larman es una versión instanciada y más específica. Este método aparece ilustrado en la Figura 43.

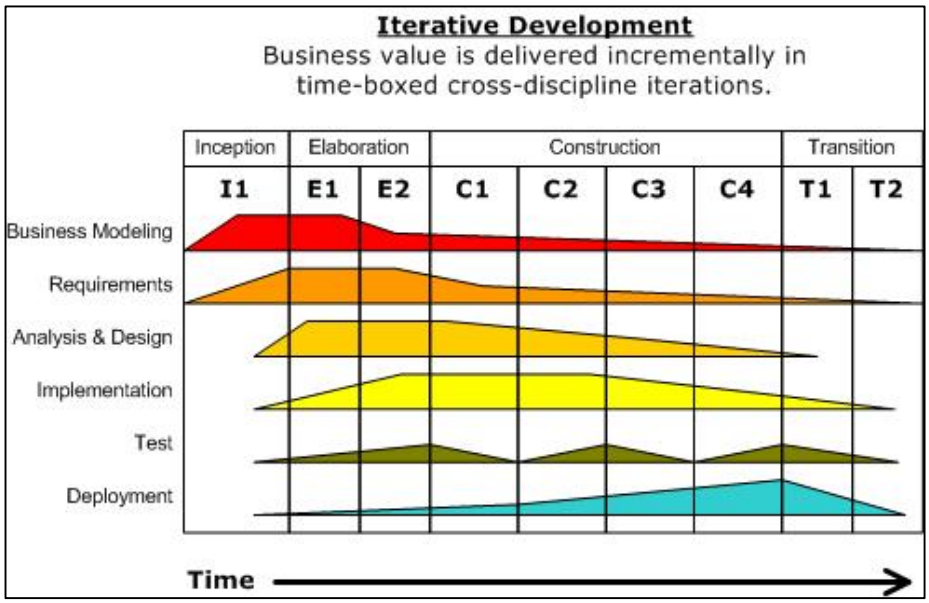

Figura 43- Proceso de Desarrollo de Software en el tiempo - Ejemplo del RUP

En la anterior imagen se aprecia la carga de trabajo que se dedica a cada tipo de actividades en el transcurso del tiempo (eje horizontal). Como se puede observar, en la fase inicial (*Inception*), el esfuerzo está enfocado en actividades correspondientes a tareas de Modelado de Negocios (y procesos) y Especificación de Requisitos.

En la siguiente fase (*Elaboration*), se continúa con estas actividades, pero comienza a trabajarse también con actividades de análisis, diseño y, paulatinamente, se comienza con la implementación y código.

En la fase de construcción el foco está en la implementación (y codificación), ciertas actividades de análisis y diseño remanentes de la fase anterior, y se comienzan las primeras tareas relativas al despliegue e instalación de la solución (producto).

La fase de transición se enfoca en el despliegue de la solución desarrollada o producto software y la realización de pruebas y validación de la solución.

Además, durante todo el proceso se realizan pruebas sobre el producto que se desarrolla o alguno(s) de su(s) componente(s), lo que contribuye a mejorar el producto y redefinir y realimentar el proceso de creación de soluciones software.

4.5.2 El funcionamiento del modelo de simulación

De manera muy similar al comportamiento del RUP mostrado en la Figura 43, el modelo de simulación aquí presentado es una representación de un proceso general del proceso software. Es de resaltar las similitudes gráficas entre la Figura 43 y la Figura 30 y Figura 32, en que se corresponden el orden de ejecución de las etapas, la flexibilidad de que sin ser totalmente asíncronas se puedan diferenciar en el transcurso del tiempo de modo que permitan tener un panorama general del proceso para alimentar criterios al observar y tratar de comprender el funcionamiento del sistema real: el proceso software.

5 Conclusiones y líneas de investigación futuras

Como producto de todo el trabajo aquí realizado, conviene dar la vuelta y preguntarse cómo ha sido el proceso de realización de esta investigación, los problemas que surgieron durante las distintas fases de la investigación que afectaron la consecución de los objetivos planteados, y las potenciales mejoras y líneas de investigación que críticamente se pueden vislumbrar a partir del trabajo aquí desarrollado. Con el fin de presentar una panorámica crítica *post-investigación*, se exponen a continuación las conclusiones y logros de forma clasificada para una mejor comprensión.

5.1 Conclusiones generales

Al principio de esta investigación lucía como el más probable panorama que se terminaría encontrando un enfoque para el modelado y la simulación claramente más útil y destacado sobre los demás; sin embargo, luego de haber abordado el estudio e investigación del estado del arte sobre los enfoques de *Eventos Discretos*, *Agentes y Sistemas Multi-agente*, y la *Dinámica de Sistemas*, aplicados al campo de la Ingeniería del Software, resulta claro que, lejos de ser excluyentes, estos enfoques son complementarios y que todos tienen valiosos aportes a la investigación. Durante la realización de este trabajo se pudo observar que, incluso en los casos en que se tomaban en cuenta aspectos de dos de estos enfoques, emergían aspectos sobre el comportamiento del sistema que resultaban de gran utilidad en la tarea de comprender el sistema real con mayor precisión, y a partir de modelos representativos, que termina siendo uno de los principales objetivos del Modelado y Simulación como campo de la ciencia en general.

En este trabajo, se hizo una recuperación extensa de artículos y publicaciones relacionadas con el Modelado y Simulación de Procesos Software. A partir de esta recuperación (cuya selección se resume en la

Tabla 1), fue posible identificar trabajos que de alguna manera han contribuido al estado actual de la cuestión, identificándose distintos paradigmas de modelado y simulación que tienen utilidad para representar los procesos software, pero de los que resaltan sobre los demás tres enfoques principalmente: la Dinámica de Sistemas, el enfoque de Agentes y Sistemas Multi-agente, y la Simulación de Eventos Discretos. Se revisaron los principales documentos seleccionados sobre estos tres enfoques con una doble intención: por una parte la intención de alimentar la perspectiva con que sería abordado el problema en sus inicios, a fin de enriquecer la variedad conceptual e interpretativa con que se enfocarían los problemas y sus posibles soluciones; y por otra parte la intención de establecer una "*fotografía*" del estado actual de la investigación de modelos de procesos software, lo que permitió dar un contexto definido a este trabajo, e identificar caminos de investigación relacionados, potencialmente aprovechables en el futuro y en las continuidades de esta investigación inicial.

La revisión de trabajos basados en estos tres enfoques, en conjunto con una evaluación crítica sobre las potencialidades de cada uno de ellos para adaptarse a las características y necesidades de las más recientes tendencias y estilos organizativos y de desarrollo de software, nos han llevado a decidir trabajar con la Dinámica de Sistemas como enfoque para la construcción del modelo que se presentó en la sección 3.5. Esto, basados en el hecho de que el proceso software, lejos de ser metódico y procedimental resulta ser complejo, variante y en muchas ocasiones contra-intuitivo.

Para la construcción del modelo, y atendiendo a la metodología de modelado y simulación tomada como referencia (Sterman, 2000); 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, que tiene definidos una serie de parámetros comunes al proceso software, que tienen incidencia en

el desarrollo del proceso y en el consumo de recursos relacionado a este proceso. 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, multi-usuario y con significativo potencial para dar soporte a la toma de decisiones en los equipos de gestión de procesos software.

La buena inmersión en el funcionamiento del proceso software y la identificación fundamentada de los parámetros que afectan el proceso, sirvió de 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 sub-proceso 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 superficial exploración de este. 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 no se deben ignorar.

En el mismo orden de ideas, han emergido de esta investigación diferentes 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. Es por ello que se propone dar continuidad al presente proyecto de fin de máster en tres criterios:

- Primero, mejorando el modelo aquí desarrollado por medio de la experimentación con ecuaciones y tablas de funciones para representar los comportamientos esperados de los subprocesos, lo que no fue posible hacer para este trabajo por límites de tiempo;
- Segundo, complementando el modelo aquí presentado con modelos alternativos desarrollados siguiendo enfoques distintos al de la Dinámica de Sistemas, lo que permitiría contrastar entre enfoques e ir definiendo criterios de diferenciación;
- Y tercero, simulando procesos que se presentan estructurados de forma distinta al modelo de Larman, tanto modelos más metódicos (y por ende irreales), como modelos más orgánicos y ágiles (por tanto más reales), para poder contrastar con fundamentos entre modelos y hacer contribuciones de peso al conocimiento sobre Modelado y Simulación de Procesos Software.

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

5.2 Puntos de mejora para trabajos consecuentes

Este trabajo ha sido una experiencia enriquecedora en muchos sentidos, entre los que merece la pena destacar:

- Al inicio de la investigación no parecía muy claro cuál sería el camino a seguir; sin embargo, sí estaba claro cuál era la meta: lograr un modelo que simulara el proceso de desarrollo de software. Al comenzar a explorar los distintos enfoques posibles para construir el modelo y las distintas estructuras de referencia, comenzó a parecer más claro no el camino a seguir sino la gama tan extensa de posibilidades, y por ende complejidades a tener en cuenta. Se optó por usar la Dinámica de Sistemas como paradigma de simulación y el Método de Larman como referencia del proceso software; sin embargo, una observación importante ahora que se ha construido el modelo, es la necesidad de contar con modelos alternativos basados en paradigmas distintos (agentes, discretos, etc.) y en modelos de proceso software distintos (RUP, V-model, etc.). Lo anterior nos permite ampliar el *background* conceptual y experimental en torno a la simulación de procesos software, además de permitirnos discernir con mejor fundamento que la visualización de la conveniencia de un método u otro, es una puerta de entrada hacia el diseño de un marco conceptual para el modelado y simulación del proceso software en el que se tengan claramente establecidos criterios para la escogencia, tanto de los paradigmas a usar, como de los modelos de proceso más convenientes según el caso lo requiera.
- Sobre este mismo modelo sería conveniente hacer exploraciones adicionales en las que se prueben elementos que tienen alta influencia en el sistema simulado, como las funciones usadas para estimar los comportamientos, los parámetros escogidos para ser usados en el modelo, o las ecuaciones seleccionadas como estimadores iniciales (las ecuaciones COCOMO).
- A pesar de que se llevó a cabo una encuesta que permitió recolectar información sobre el proceso software con la intención de calibrar el modelo con los valores que por defecto tendría el sistema, resultó que habían discrepancias importantes

entre las respuestas de los encuestados, a pesar de que todos tenían un nivel considerable de experimentación en el área, lo que nos abre a la posibilidad de concebir una futura investigación en la que con un buen diseño estadístico se difunda un cuestionario a un número mayor de encuestados con la intención de disminuir las discrepancias que se presentan cuando el número de encuestados es pequeño, y así lograr una muestra significativa y con un alto grado de confianza como para calibrar no sólo este modelo sino los que se desarrollen bajo distintos paradigmas que le sirvan a éste de complemento.

5.3 Otros caminos futuros

Aunque un poco ambicioso, ha surgido de esta investigación la curiosidad por indagar sobre las posibilidades de, en paralelo al diseño del marco conceptual para el modelado y simulación de procesos software, concebir la posibilidad de la creación de un software que permita de un modo más "amigable hacia el usuario" configurar parámetros, valores y ecuaciones, con la intención de que el proceso de modelado y simulación esté un poco menos restringido a personal altamente cualificado en el campo matemático e ingenieril y sea más utilizable por perfiles no tan técnicos; con lo que el potencial de la herramienta se multiplicaría y abriría excelentes posibilidades tanto al mundo académico como al empresarial.

Finalmente, para cerrar con las opiniones de conclusión de este trabajo, parece importante la sugerencia de incluir los diferentes enfoques de "Modelado y Simulación" en los programas formativos tanto de pregrado como de postgrado, fundamentado en que lejos de perder vigencia, con el acelerado crecimiento de las complejidades de la humanidad y con los cada vez más potentes procesadores y capacidades informáticas, el modelado y simulación de sistemas parece ser una excelente, económica, multifacética, y ajustable herramienta para dar soporte a la toma de decisiones a distintos niveles organizativos.

6 Planificación del trabajo

A continuación se presenta un panorama en forma de esquema que resume las distintas actividades que fueron necesarias para llevar a cabo el proceso de creación de este trabajo, y la representación de su carga en el espacio temporal durante todo el proceso de investigación.

Para lograr convertir lo que fue la idea inicial que motivó este trabajo, en el modelo construido y este trabajo mismo, fue necesario definir una serie de actividades que debían ser atendidas para, en conjunto, derivar en un trabajo sólido, revisado, que es lo que se ha pretendido entregar aquí. Veamos estas actividades en la Figura 44:

Nombre	Fecha de inicio	Fecha de fin
Conceptualización del Problema	22/09/14	3/02/15
Investigación Preliminar sobre Modelado y Simulación de Procesos Software	22/09/14	20/11/14
Realización de la Revisión Sistemática	1/10/14	3/02/15
Investigación en detalle	10/02/15	22/06/15
Investigación del Estado del Arte	10/02/15	22/06/15
Implementación de la propuesta	1/05/15	1/09/15
Definición del Diagrama Causal	13/08/15	13/08/15
Diseño en VENSIM	1/05/15	1/09/15
Realización de pruebas preliminares	20/07/15	1/09/15
Estudio de escenarios	7/08/15	7/08/15
Pruebas del modelo	13/08/15	3/09/15
Diseño de hipótesis que sirvan para validar el modelo	13/08/15	3/09/15
Realización de Pruebas	21/08/15	3/09/15
Reuniones (on y offline), tutorías	24/11/14	31/08/15
Escritura de la memoria	9/02/15	7/09/15

Figura 44 - Detalle de las actividades llevadas a cabo.

Algunas tareas o actividades llevadas a cabo tienen que ver con la definición misma de la idea que sería *a posteriori* el hilo conductor de todo el proceso de realización del trabajo de fin de máster; a este conjunto de actividades se le ha llamado *Conceptualización del Problema*. En la Figura 44 se observa que se han listado las actividades:

- Investigación Preliminar sobre Modelado y Simulación de Proceso Software.
- Realización de la Revisión Sistemática.

Al trabajo de búsqueda de artículos y bibliografía relacionada con el modelado y simulación del proceso software, ya con objetivos definidos y delimitados, se le ha llamado *Investigación Conceptual*. El detalle de la actividad en la Figura 44, se muestra como:

- Investigación del Estado del Arte.

Al conjunto de actividades que concierne la creación, experimentación, "juego" y configuraciones del modelo de simulación creado, se les denominará *Implementación de la propuesta*. Sus actividades son:

- Definición del diagrama causal.
- Diseño en VENSIM.
- Realización de pruebas preliminares.
- Estudio de escenarios.

Al conjunto de actividades que conciernen la manipulación y prueba del modelo se le ha llamado en el diagrama: *Pruebas del modelo*, y sus actividades son:

- Diseño de hipótesis que sirvan para validar el modelo.
- Realización de pruebas.

Como parte de las actividades de supervisión y seguimiento por parte de las profesoras tutoras, se realizaron una serie de encuentros-reuniones con doble finalidad: mostrar a las tutoras los avances que se iban logrando con respecto a la investigación y recibir por parte de ellas su orientación y re-direccionamiento en base a las fortalezas, debilidades y oportunidades de mejora e investigación que pudieran detectar en torno al trabajo realizado. Este conjunto de encuentros se ha definido en el diagrama que sigue como:

- Reuniones (on y off-line), tutorías.

A la actividad de plasmar con palabras toda la experiencia académica en un documento, darle formato, revisar y corregir referencias bibliográficas, se le denominó:

- Elaboración de la Memoria.

Las actividades se llevaron a cabo en los tiempos y carga de trabajo que se ven en Figura 44 y Figura 45.

Además, en la Figura 45, es posible visualizar gráficamente la ejecución en el tiempo del trabajo de investigación plasmado en este escrito. Como es de esperar, hay algunas actividades que son precedentes a otras, sin embargo, lo que llama la atención es el poco tiempo que tomó la construcción del modelo en comparación con todo el tiempo dedicado a la conceptualización, delimitación y definición del modelo a construir.

Aquí es relevante mencionar el carácter continuo que tuvo la relación documento-modelo como consecuencia de las múltiples versiones del modelo que se probaron. Cada vez que el modelo era mejorado requería cambios en el documento con la finalidad de representar las mejoras identificadas que hacían más representativo el modelo y por ende más útil a la hora de establecer líneas futuras de investigación o propuestas para modelos alternativos que permitan contrastar la representatividad y utilidad del modelo.

Figura 45 - Diagrama de Gantt de la realización del trabajo

Bibliografía

Agarwal, R., & Umphress, D. (2010). A flexible model for simulation of software development process. *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10*, 1. doi:10.1145/1900008.1900064

Boehm, B. (2012). COCOMO II Model Definition Manual. *The American Journal of Tropical Medicine and Hygiene*, *87*(5 Suppl), i. doi:10.4269/ajtmh.2012.875suppack

California., U. of S. (2003). USC COCOMO II: User´s manual. In *University of Southern California.* (p. 93).

Cao, L. a N. (2010). Modeling Dynamics in Agile Software Development. *ACM Transactions on Management Information Systems (TMIS)*, *1*(1), 5–26. doi:10.1145/1877725.1877730

Cherif, R., & Davidsson, P. (2009). Software Development Process Simulation: Multi Agent-Based Simulation versus System Dynamics (p. 12). Retrieved from http://download.springer.com.strauss.uc3m.es:8080/static/pdf/168/bok%253A978-3-642-13553-8.pdf?auth66=1416753949_a627175ef63f04d5511cbebdbdfbad33&ext=.pdf

Cherif, R., & Davidsson, P. (2010). Software development process simulation: Multi agent-based simulation versus system dynamics. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5683 LNAI*, 73–85. doi:10.1007/978-3-642-13553-8_7

De Sousa Coelho, J. J., Braga, J. L., & Ambrósio, B. G. (2013). System dynamics model for simulation of the software inspection process. *ACM SIGSOFT Software Engineering Notes*, *38*(5), 1. doi:10.1145/2507288.2507306

Donzelli, P., & Iazeolla, G. (2000). Hybrid Simulation Modelling of the Software Process, *59*, 1–13.

Ferreira, S., Collofello, J., Shunk, D., & Mackulak, G. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *Journal of Systems and Software*, *82*(10), 1568–1577. doi:10.1016/j.jss.2009.03.014

Filho, R. C. S., & Rocha, a. R. C. Da. (2010). Towards an Approach to Support Software Process Simulation in Small and Medium Enterprises. *Software*

Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on. doi:10.1109/SEAA.2010.67

García, J. M. (2006). *Teoría y ejercicios prácticos de Dinámica de Sistemas*.

Izquierdo, L. R. (Universidad de B. (2008). Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. *Revista de Metodologia de Ciencias Sociales*, *16*(16), 85–112. Retrieved from http://e-spacio.uned.es/fez/eserv.php?pid=bibliuned:Empiria-2008-16-10575&dsID=PDF

Kellner, M. I., Madachy, R. J., & Raffo, D. M. (1999). Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, *46*(2-3), 91–105. doi:10.1016/S0164-1212(99)00003-5

Khosrovian, K., Pfahl, D., & Garousi, V. (2008). GENSIM 2 . 0 : A Customizable Process Simulation Model, 294–306. Retrieved from http://download.springer.com.strauss.uc3m.es:8080/static/pdf/716/chp%253A10.1007%252F978-3-540-79588-9_26.pdf?auth66=1416869561_1f73788e7be297c950765afa7e336127&ext=.pdf

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Prentice Hall.

Martin, R., & Raffo, D. (2001). Application of a hybrid process simulation model to a software development project. *Journal of Systems and Software*, *59*, 237–246. doi:10.1016/S0164-1212(01)00065-6

Park, S., & Bae, D.-H. (2011). An approach to analyzing the software process change impact using process slicing and simulation. *Journal of Systems and Software*, *84*(4), 528–543. doi:10.1016/j.jss.2010.11.919

Park, S., Choi, K., Yoon, K., & Bae, D.-H. (2008). Deriving Software Process Simulation Model from SPEM-based Software Process Model. *14th Asia-Pacific Software Engineering Conference (APSEC'07)*, 382–389. doi:10.1109/ASPEC.2007.28

Pressman, R. S. (2010). *Software Engineering: A Practitioner´s Approach* (7th ed.). McGraw-Hill.

Raffo, D. (2012). Process simulation will soon come of age: Where's the party? *2012 International Conference on Software and System Process (ICSSP)*, *1*, 231–231. doi:10.1109/ICSSP.2012.6225975

Raffo, D., Vandeville, J., & Martin, R. (1999). Software Process Simulation to Achieve Higher CMMLevels, *46*, 163–172.

Ramon, O., & Villegas, T. (2001). Emergent Tendencies in Multi-Agent-based Simulations using Constraint-based Methods to Effect Practical Proofs over Finite Subsets of Simulation Outcomes Table of Contents. *Facilities*, *PhD*, 296. Retrieved from http://cfpm.org/cpmrep86.html

Robertson, S. (2005). Learning from Other Disciplines.

Rodríguez, D., Ruiz, M., Riquelme, J. C., & Harrison, R. (2011). Multiobjective simulation optimisation in software project management. *Genetic and Evolutionary Computation Conference, GECCO'11*, 1883–1890. doi:10.1145/2001576.2001829

Rus, I., Collofello, J., & Lakey, P. (1999). Software process simulation for reliability management. *Journal of Systems and Software*, *46*, 173–182. doi:10.1016/S0164-1212(99)00010-2

Rus, I., Neu, H., & Munch, J. (2003). A systematic methodology for developing discrete event simulation models of software development processes, (ProSim). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.3840&rep=rep1&type=pdf

Saoud, N. B.-B. (2002). Agent-based approach for software development process simulation. *IEEE SMC WA1Q1*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1175572

Smith, C. (1991). Improving Service While Controlling Costs. *IEEE Software*, *March*, 95–96.

Spasic, B., & Onggo, B. S. S. (2012). Agent-based simulation of the software development process: a case study at AVL, (1999). Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6465117

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

Ventana Systems. (2011). VENSIM. Ventana Systems. Retrieved from http://www.vensim.com/download.html

Vicente, R. (2012a). Modelamiento semántico con Dinámica de Sistemas en el proceso de desarrollo de software. *Iberian Journal of Information Systems and Technologies*, (10), 19–33. doi:10.4304/risti.10.19-34

Vicente, R. (2012b). Modelamiento semántico con Dinámica de Sistemas en el proceso de desarrollo de software. *Iberian Journal of Information Systems and Technologies*, (10), 19–33. doi:10.4304/risti.10.19-34

Wu, M., & Yan, H. (2009). Simulation in software engineering with system dynamics: A case study. *Journal of Software*, *4*(10), 1127–1135. doi:10.4304/jsw.4.10.1127-1135

Yu, W. D. (1990). A Modeling Approach to Software Cost Estimation. *IEEE J. Select. Areas Communications*, *8*, 309–314.

Zhang, H., Jeffery, R., Houston, D., Huang, L., & Zhu, L. (2011). Impact of process simulation on software practice: an initial report. *Software Engineering (ICSE), 2011 33rd International Conference on*, 1046–1056. doi:10.1145/1985793.1985993

Zhang, H., Kitchenham, B., & Jeffery, R. (2007). A Framework for Adopting Software Process Simulation in CMMI Organizations, 320–331.

Zhang, H., Kitchenham, B., & Pfahl, D. (2008). Software Process Simulation Modeling: Facts, Trends and Directions. *2008 15th Asia-Pacific Software Engineering Conference*, 59–66. doi:10.1109/APSEC.2008.50

Zhang, H., Raffo, D., Birkhöltzer, T., Houston, D., Madachy, R., Münch, J., & Jr, S. M. S. (2014). Software process simulation — at a crossroads ?, (October), 923–928. doi:10.1002/smr

Anexos

A. Informe de ecuaciones del modelo en VENSIM.

(001) "%GLana"=
0.2
Units: **undefined**

(002) "%GLdes"=
0.1
Units: **undefined**

(003) "%GLimp"=
0.005
Units: **undefined**

(004) "%GLtest"=
0.005
Units: **undefined**

(005) "%PMAna"=
0.4
Units: **undefined**

(006) "%PMDes"=
0.2
Units: **undefined**

(007) "%PMImp"=
0.3
Units: **undefined**

(008) "%PMTest"=
0.1
Units: **undefined**

(009) "%TimeAna"=
ANtime/TPtime
Units: **undefined**

(010) "%TimeCons"=
COtime/TPtime
Units: **undefined**

(011) "%TimeDe"=
DEtime/TPtime
Units: **undefined**

(012) "%TimeTest"=
TEtime/TPtime
Units: **undefined**

(013) A=
1.01
Units: **undefined**

(014) ACAP=
0.71
Units: **undefined**

(015) AEXP=
0.82
Units: **undefined**

(016) AggregatedTime=
ANtime+DEtime+COtime+TEtime
Units: **undefined**

(017) analisis=
IF THEN ELSE(PM Analisis>0, (((("RU-D-A"/100))*design)+((("RU-I-A"/100))
*implementation)+((1+("RU-T-A"/100))*tests))*analisis rate*(ACAP*AEXP*SCED
*TOOL))/4, 0)
Units: **undefined**

(018) Analisis aggregation= INTEG (
analisis,
0)
Units: **undefined**

(019) analisis behavior(
[(0,-0.2)-(50,1)],(0,0),(0.152905,0.1),(2.44648,0.135965),(6.11621,0.614035
),(9.48012,0.811404),(22.1713,0.785088),(27.37,0.337719),(30.7339,0.166667
),(35.6269,0.0745614),(43.578,0.0394737),(48.7768,0.0219298),(49,0.00438596
),(50,0))
Units: **undefined**

(020) Analisis effort use=
PM Analisis*normal analisis effort use rate*0.5
Units: **undefined**

(021) analisis rate=
analisis behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime)
)))*50)
Units: **undefined**

(022) ANPeopleCost=
PMana/("TDEV-PMadjPA"*"%TimeAna")
Units: **undefined**

(023) ANtime=
IF THEN ELSE(Relative Complexity=1, 0.46, IF THEN ELSE(Relative Complexity
=2, 0.83, IF THEN ELSE(Relative Complexity=3, 1.15, IF THEN ELSE(Relative Complexity
=4, 1.78, 2.5))))
Units: **undefined**

(024) B=

A+0.01*(FLEX+PMAT+PREC+RESL+TEAM)

Units: **undefined**

(025) Code= INTEG (
implementation,
0)
Units: **undefined**

(026) COPeopleCost=
PMimp/("TDEV-PMadjPA"*"%TimeCons")
Units: **undefined**

(027) COtime=
IF THEN ELSE(Relative Complexity=1, 1.1, IF THEN ELSE(Relative Complexity
=2, 1.9, IF THEN ELSE(Relative Complexity=3, 3.48, IF THEN ELSE(Relative Complexity
=4, 4.8, 9.2))))
Units: **undefined**

(028) CPLX=
1
Units: **undefined**

(029) DATA=
1
Units: **undefined**

(030) DEPeopleCost=
PMdes/("TDEV-PMadjPA"*"%TimeDe")
Units: **undefined**

(031) design=
IF THEN ELSE(PM Design>0, DELAY3((tests*("RU-T-D"/100)+implementation*(
"RU-I-D"/100))*design rate*(AEXP*CPLX*SCED*TOOL), ANtime), 0)
Units: **undefined**

(032) Design aggregation= INTEG (
design,
0)
Units: **undefined**

(033) design behavior(
[(0,0)-(50,1)],(0,0),(7.95107,0.05),(11.0092,0.1289),(14.526,0.600877),(22.63
,0.714912),(27.37,0.697368),(31.3456, 0.596491),(34.4037,0.276316),(35.6269
,0.114035),(39.6024,0.0745614),(44.3425,0.0307018),(49.8471,0))
Units: **undefined**

(034) Design effort use=
IF THEN ELSE(PM Design>0:AND:PM Analisis<PMana*0.25, DELAY3(0.5*PM Design
*normal design effort rate, ANtime*0.8), 0)
Units: **undefined**

(035) design rate=
design behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime)))
)*50)
Units: **undefined**

(036) DEtime=
IF THEN ELSE(Relative Complexity=1, 0.65, IF THEN ELSE(Relative Complexity
=2, 1.18, IF THEN ELSE(Relative Complexity=3, 1.62, IF THEN ELSE(Relative Complexity
=4, 2.3, 3.03))))
Units: **undefined**

(037) DOCU=
1
Units: **undefined**

(038) FCIL=
(SITE+TOOL)/2
Units: **undefined**

(039) FINAL TIME = 100
Units: Month
The final time for the simulation.

(040) FLEX=
5
Units: **undefined**

(041) GLContribution=

("%GLana"*analisis+"%GLdes"*design+"%GLimp"*implementation+"%GLtest"*tests
)*ACAP*AEXP*SCED*TOOL
Units: **undefined**

(042) Glossary aggregation= INTEG (
GLContribution,
0)
Units: **undefined**

(043) implementation=
IF THEN ELSE(PM Implementation>0, 100*DELAY3((("RU-T-I"/100)+1)*tests)+1000
*implementation rate*AEXP*CPLX*DATA*LEXP*MODP*PCAP*RELY

*SCED*STOR*TIMEp*TOOL*TURN
*VEXP
*VIRT,
(ANtime+DEtime)*0.5), 0)
Units: **undefined**

(044) implementation behavior(
[(-1,0)-(50,1)],(0,0),(3.0581,0.0175439),(5.04587,0.0745614),(6.95413,0.236842),(8.82569,0.267544),(11.1651,0.135965),(13.9725,0.0921053),(16.3119,0.100877),(20.8349,0.320175),(23.9541,0.736842),(30.1927,0.907895),(31.7523,0.899123),(32.844,0.714912),(36.2752,0.548246),(37.8349,0.438596),(38.6147,0.276316),(43.9174,0))
Units: **undefined**

(045) implementation effort use=
IF THEN ELSE(PM Implementation>0, DELAY3(0.3*PM Implementation*normal implementation effort rate
, (ANtime+DEtime)), 0)
Units: **undefined**

(046) implementation rate=
implementation behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime))))*50)
Units: **undefined**

(047) INITIAL TIME = 0
Units: Month
The initial time for the simulation.

(048) LEXP=
1
Units: **undefined**

(049) LOC=
IF THEN ELSE(Relative Complexity=1, 5000, IF THEN ELSE(Relative Complexity
=2, 11500, IF THEN ELSE(Relative Complexity=3, 18900, IF THEN ELSE(Relative Complexity
=4, 44000, 1.08e+006))))
Units: **undefined**

(050) LTEX=
0.9
Units: **undefined**

(051) MODP=
1
Units: **undefined**

(052) NED=
PMadjED/"TDEV-PMadjED"
Units: **undefined**

(053) normal analisis effort use rate=
analisis behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime)
)))*50)
Units: **undefined**

(054) normal design effort rate=
design behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime)))
)*50)
Units: **undefined**

(055) normal implementation effort rate=
implementation behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime
))))*50)
Units: **undefined**

(056) normal test effort rate=
test behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime))))*
50)
Units: **undefined**

(057) NPA=
PMadjPA/"TDEV-PMadjPA"
Units: **undefined**

(058) PCAP=
0.7
Units: **undefined**

(059) PCON=
1
Units: **undefined**

(060) PDIF=
(PVOL+STOR+TIMEp)/3
Units: **undefined**

(061) People Related Cost=
NPA*PPP
Units: **undefined**

(062) PERS=
(ACAP+PCAP+PCON)/3
Units: **undefined**

(063) PEXP=
1
Units: **undefined**

(064) PM Analisis= INTEG (
-Analisis effort use,
PMana)
Units: **undefined**

(065) PM Design= INTEG (
-Design effort use,
PMdes)
Units: **undefined**

(066) PM Implementation= INTEG (
-implementation effort use,
PMimp)
Units: **undefined**

(067) PM TEsts= INTEG (
-Test effort use,
PMtest)
Units: **undefined**

(068) PMadjED=

PMnom*(FCIL*PDIF*PERS*PREX*RCPX*RUSE*SCED)
Units: **undefined**

(069) PMadjPA=

PMnom*(ACAP*AEXP*CPLX*DATA*DOCU*LTEX*PCAP*PCON*PEXP*PVOL*RELY*RUSE*SCED*SITE
*STOR*TIMEp*TOOL)
Units: **undefined**

(070) PMana=
PMadjPA*"%PMAna"
Units: **undefined**

(071) PMAT=
5
Units: **undefined**

(072) PMdes=
PMadjPA*"%PMDes"
Units: **undefined**

(073) PMimp=
"%PMImp"*PMadjPA
Units: **undefined**

(074) PMnom=
SIZE^B
Units: **undefined**

(075) PMtest=
PMadjPA*"%PMTest"
Units: **undefined**

(076) PPP=
4500
Units: **undefined**

(077) PREC=
5
Units: **undefined**

(078) PREX=
(AEXP+LTEX+PEXP)/3
Units: **undefined**

(079) PVOL=
1
Units: **undefined**

(080) QOP=
IF THEN ELSE(Relative Complexity=1, 3, IF THEN ELSE(Relative Complexity=2
, 6, IF THEN ELSE(Relative Complexity=3, 10.2, IF THEN ELSE(Relative Complexity
=4, 13.5, 57))))
Units: **undefined**

(081) QUC=
IF THEN ELSE(Relative Complexity=1, 12, IF THEN ELSE(Relative Complexity=
2, 20, IF THEN ELSE(Relative Complexity=3, 29.5, IF THEN ELSE(Relative Complexity
=4, 41, 73))))
Units: **undefined**

(082) RCPX=

(CPLX+DATA+DOCU+RELY)/4
Units: **undefined**

(083) Relative Complexity=
5
Units: **undefined**

(084) RELY=
1
Units: **undefined**

(085) RESL=
0
Units: **undefined**

(086) "RU-D-A"=
IF THEN ELSE(Relative Complexity=1, 30, IF THEN ELSE(Relative Complexity=
2, 32, IF THEN ELSE(Relative Complexity=3, 47, IF THEN ELSE(Relative Complexity
=4, 48, 64))))
Units: **undefined**

(087) "RU-I-A"=
IF THEN ELSE(Relative Complexity=1, 11, IF THEN ELSE(Relative Complexity=
2, 16.9, IF THEN ELSE(Relative Complexity=3, 17.5, IF THEN ELSE(Relative Complexity
=4, 29, 34))))
Units: **undefined**

(088) "RU-I-D"=
IF THEN ELSE(Relative Complexity=1, 12, IF THEN ELSE(Relative Complexity=
2, 14.5, IF THEN ELSE(Relative Complexity=3, 18.5, IF THEN ELSE(Relative Complexity
=4, 29, 33))))
Units: **undefined**

(089) "RU-T-A"=

IF THEN ELSE(Relative Complexity=1, 13.5, IF THEN ELSE(Relative Complexity
=2, 15.5, IF THEN ELSE(Relative Complexity=3, 24, IF THEN ELSE(Relative Complexity
=4, 34, 37))))
Units: **undefined**

(090) "RU-T-D"=
IF THEN ELSE(Relative Complexity=1, 10, IF THEN ELSE(Relative Complexity=
2, 13, IF THEN ELSE(Relative Complexity=3, 19.5, IF THEN ELSE(Relative Complexity
=4, 22.5, 34))))
Units: **undefined**

(091) "RU-T-I"=
IF THEN ELSE(Relative Complexity=1, 7.5, IF THEN ELSE(Relative Complexity
=2, 11.5, IF THEN ELSE(Relative Complexity=3, 17, IF THEN ELSE(Relative Complexity
=4, 22.5, 31.5))))
Units: **undefined**

(092) RUSE=
1
Units: **undefined**

(093) SAVEPER =
TIME STEP
Units: Month [0,?]
The frequency with which output is stored.

(094) SCED=
1.1
Units: **undefined**

(095) SITE=
1
Units: **undefined**

(096) SIZE=
200
Units: **undefined**

(097) STOR=
1
Units: **undefined**

(098) "TDEV-PMadjED"=

(3.67*(PMadjED^(0.28+0.2*(B-1.01))))*(SCED*100/100)
Units: **undefined**

(099) "TDEV-PMadjPA"=

(3.67*(PMadjPA^(0.28+0.2*(B-1.01))))*(SCED*100/100)
Units: **undefined**

(100) "TDEV-PMnom"=
(3.67*(PMnom^(0.28+0.2*(B-1.01))))*(SCED*100/100)
Units: **undefined**

(101) TEAM=
5
Units: **undefined**

(102) TEPeopleCost=
PMtest/("TDEV-PMadjPA"*"%TimeTest")
Units: **undefined**

(103) test behavior(
[(0,0)-(50,1)],(0,0.00438596),(0.458716,0.0526316),(1.68196,0.0526316),(3.51682
,0.0657895),(5.19878,0.0657895),(5.50459,0.0657895),(6.42202,0.0570175),(7.64526
,0.0394737),(8.56269,0.0394737),(8.8685,0.0394737),(11.0092,0.0219298),(11.6208
,0.0219298),(13.6086,0.0438596),(13.9144,0.0614035),(14.8318,0.0964912),(15.5963
,0.127193),(16.6667,0.149123),(17.2783,0.157895),(20.4893,0.162281),(20.948
,0.162281),(23.3945,0.157895),(24.7706,0.131579),(26.7584,0.0964912),(28.8991
,0.0964912),(29.8165,0.0745614),(31.9572,0.0438596),(33.3333,0.114035),(35.0153
,0.245614),(36.0856,0.355263),(37.6147,0.469298),(37.9205,0.52193),(38.8379
,0.653509),(40.5199,0.70614),(43.7309,0.692982),(45.5658,0.583333),(47.8593
,0.421053),(49.2355,0.346491),(49.8471,0.307018),(50,0))
Units: **undefined**

(104) Test effort use=
IF THEN ELSE(PM TEsts>0, DELAY3(0.2*PM TEsts*normal test effort rate, (ANtime
+COtime+DEtime)), 0)
Units: **undefined**

(105) test rate=
test behavior((Time/(MAX(AggregatedTime, MAX("TDEV-PMadjPA",TPtime))))*
50)
Units: **undefined**

(106) tests=
IF THEN ELSE(PM TEsts>0, DELAY3(100*test rate*AEXP*LEXP*MODP*PCAP*SCED*TOOL
*VEXP, ANtime+COtime+DEtime), 0)
Units: **undefined**

(107) Tests aggregation= INTEG (
tests,
0)
Units: **undefined**

(108) TEtime=
IF THEN ELSE(Relative Complexity=1, 0.4, IF THEN ELSE(Relative Complexity
=2, 0.55, IF THEN ELSE(Relative Complexity=3, 0.7, IF THEN ELSE(Relative Complexity
=4, 0.99, 1.45))))
Units: **undefined**

(109) TIME STEP = 0.0625
Units: Month [0,?]
The time step for the simulation.

(110) TIMEp=
1
Units: **undefined**

(111) TMAX=
200
Units: **undefined**

(112) TOOL=
0.83
Units: **undefined**

(113) TPtime=
IF THEN ELSE(Relative Complexity=1, 1.75, IF THEN ELSE(Relative Complexity
=2, 3.24, IF THEN ELSE(Relative Complexity=3, 6, IF THEN ELSE(Relative Complexity
=4, 10.2, 18))))
Units: **undefined**

(114) TURN=
1
Units: **undefined**

(115) VEXP=
1
Units: **undefined**

(116) VIRT=
1
Units: **undefined**

B. Encuesta realizada para calibrar el modelo.

Encuesta.

Tema: Proceso de Desarrollo de Software - Valores de parámetros.

Justificación: La siguiente "encuesta" es parte de un experimento que lleva a cabo el estudiante German Lenin Dugarte Peña, quien hace su Trabajo de Fin de Máster, del programa en Ciencia y Tecnología Informática de la Universidad Carlos III de Madrid. Se agradece inmensamente su valiosa colaboración y apoyo, al tomarse unos minutos en leer y responder.

Contexto: Como parte de la calibración de un modelo de simulación de procesos de desarrollo de software, se requiere una muestra de los valores que personas inmersas en el área de Ingeniería del Software puedan estimar para ciertos parámetros del modelo.

Encuesta:

Por favor, responda a las siguientes preguntas con los valores más exactos que pueda estimar:

1. Indique un valor numérico, **en miles de líneas de código**, que usted considere se corresponde con la complejidad de un Proyecto de desarrollo de software.

Sencillo	Poco complejo	Medio Complejo	Muy Complejo	Extremadamente Complejo

2. Indique un valor numérico, para la cantidad de personas que intervienen en un Proyecto de desarrollo de software, de acuerdo a su complejidad.

Sencillo	Poco complejo	Medio Complejo	Muy Complejo	Extremadamente Complejo

3. Indique la **cantidad de Casos de Uso** que usted diría suelen ser frecuentes en Proyectos de Desarrollo de Software, de acuerdo su grado de complejidad.

Sencillo	Poco complejo	Medio Complejo	Muy Complejo	Extremadamente Complejo

4. Indique el tiempo aproximado que suele tomar el **Proyecto TOTAL de Desarrollo de Software** (por lo menos en la mayoría de los casos), de acuerdo a su grado de complejidad. POR FAVOR INDIQUE LAS UNIDADES QUE TOMA COMO REFERENCIA (DIAS, MESES, AÑOS, ETC)

Sencillo	Poco complejo	Medio Complejo	Muy Complejo	Extremadamente Complejo

5. Indique el tiempo aproximado que suele tomar la fase formal de **Análisis** (por lo menos en la mayoría de los casos), de acuerdo a su grado de complejidad. POR FAVOR INDIQUE LAS UNIDADES QUE TOMA COMO REFERENCIA (DIAS, MESES, AÑOS, ETC)

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

6. Indique el tiempo aproximado que suele tomar la fase formal de **DISEÑO** (por lo menos en la mayoría de los casos), de acuerdo a su grado de complejidad. POR FAVOR INDIQUE LAS UNIDADES QUE TOMA COMO REFERENCIA (DIAS, MESES, AÑOS, ETC)

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

7. Indique el tiempo aproximado que suele tomar la fase formal de **CODIFICACIÓN** (por lo menos en la mayoría de los casos), de acuerdo a su grado de complejidad. POR FAVOR INDIQUE LAS UNIDADES QUE TOMA COMO REFERENCIA (DIAS, MESES, AÑOS, ETC)

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

8. Indique el tiempo aproximado que suele tomar la fase formal de **PRUEBAS** (por lo menos en la mayoría de los casos), de acuerdo a su grado de complejidad. POR FAVOR INDIQUE LAS UNIDADES QUE TOMA COMO REFERENCIA (DIAS, MESES, AÑOS, ETC)

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

Si se asume que en la fase de construcción de un Proyecto de Desarrollo de Software las fases son

Análisis>Diseño>Codificación>RealizaciónDePruebas

Y las letras A, B, C, D, E, F de la siguiente imagen representan flujo de trabajo que se **re-hace**, en un proyecto de software.

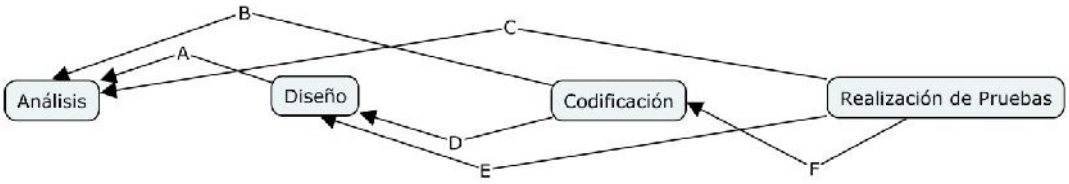

Figura 46 - Encuesta. Flujos de trabajo que se re-hace en un proceso software.

Por favor responda:

A. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, **después del Diseño** necesita ser **Re-Analizado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

B. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, después de la **Codificación** necesita ser **Re-Analizado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

C. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, después de la **Realización de las pruebas** necesita ser **Re-Analizado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

D. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, después de la **Codificación** necesita ser **Re-Diseñado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

E. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, después de la **Realización de las pruebas** necesita ser **Re-Diseñado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

F. Indique la **proporción del flujo de trabajo** que, bajo su apreciación, después de la **Realización de las pruebas** necesita ser **Re-Codificado**, en cada caso:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

Por favor indique:

1. Basado en su experiencia, ¿suele incorporar la creación de un glosario, como una tarea extra de sus proyectos de software?:

Si	No

4.1 En caso positivo, cuántos términos considera que suele tener **el glosario** según el grado de complejidad del Proyecto de Desarrollo de Software:

Sencillo	**Poco complejo**	**Medio Complejo**	**Muy Complejo**	**Extremadamente Complejo**

Opcional: Nombre y Apellidos:_____.

Experiencia:

Muy poca (<1año)	**Poca (>1,<3)**	**Media (>3,<5)**	**Alta (>5,<8)**	**Experto (>8años)**

Si tiene alguna sugerencia o comentario, puede dirigirla al estudiante al correo: germanlenin.dugarte@uc3m.es.

C. Resultados obtenidos de la encuesta realizada para calibrar el modelo.

Tabla 19 - Recopilación de resultados de la encuesta realizada para calibrar el modelo de simulación.

	Indique un valor numérico, en miles de líneas de código, que usted considere se corresponde con la complejidad de un Proyecto de desarrollo de software.							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	5000	5000	5000	500	5000	4100	500	5000
Poco Complejo	7000	50000	7000	1000	8000	14600	1000	50000
Medio Complejo	10000	1E+05	9000	3000	12000	26800	3000	100000
Muy Complejo	20000	3E+05	10000	5000	25000	72000	5000	300000
Extremadamente Complejo	50000	5E+05	2E+06	15000	100000	533000	15000	2000000
	Indique un valor numérico, para la cantidad de personas que usted estima intervienen en un Proyecto de desarrollo de software de acuerdo a su complejidad.							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	1	2	4	1	3	2,2	1	4
Poco Complejo	2	2	8	2	7	4,2	2	8
Medio Complejo	3	3	15	5	9	7	3	15
Muy Complejo	3	5	20	10	12	10	3	20
Extremadamente Complejo	5	10	100	20	25	32	5	100
	Indique la cantidad de Casos de Uso que usted diría suelen ser frecuentes en Proyectos de Desarrollo de Software de acuerdo su grado de complejidad.							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	10	5	15	1	10	8,2	1	15
Poco Complejo	15	15	25	3	15	14,6	3	25
Medio Complejo	20	30	35	5	25	23	5	35
Muy Complejo	25	40	50	7	35	31,4	7	50
Extremadamente Complejo	30	50	100	10	60	50	10	100
	Indique el tiempo aproximado que suele tomar el Proyecto TOTAL de Desarrollo de Software, de acuerdo a su grado de complejidad. Por favor indique las unidades que toma como referencia (horas, dias, semanas, meses, etc).							
	Enc.	**Enc.**	**Enc.**	**Enc.**	**Enc.**	**Media**	**Mini**	**Máxi**

	1	2	3	4	5		mo	mo
Sencillo	3	3	1	0,75	1	1,75	0,75	3
Poco Complejo	6	5	2	1,2	2	3,24	1,2	6
Medio Complejo	8	10	4	2	6	6	2	10
Muy Complejo	12	20	6	4	9	10,2	4	20
Extremadamente Complejo	24	30	12	12	12	18	12	30

	Indique el tiempo aproximado que suele tomar la fase formal de Análisis en un Proyecto de Desarrollo de Software, de acuerdo a su grado de complejidad. Por favor indique las unidades que toma como referencia (horas, dias, semanas, meses, etc).							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	0,8	0,5	0,25	0,25	0,5	0,46	0,25	0,8
Poco Complejo	1,5	0,65	0,75	0,5	0,75	0,83	0,5	1,5
Medio Complejo	2	1	1	0,75	1	1,15	0,75	2
Muy Complejo	3	2	1,3	1,3	1,3	1,78	1,3	3
Extremadamente Complejo	4	3	1,5	2	2	2,5	1,5	4

	Indique el tiempo aproximado que suele tomar la fase formal de Diseño en un Proyecto de Desarrollo de Software, de acuerdo a su grado de complejidad. Por favor indique las unidades que toma como referencia (horas, dias, semanas, meses, etc).							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	1	1	0,5	0,25	0,5	0,65	0,25	1
Poco Complejo	1,7	2	1	0,5	0,7	1,18	0,5	2
Medio Complejo	2	2,5	1,3	1	1,3	1,62	1	2,5
Muy Complejo	3	3	1,5	2	2	2,3	1,5	3
Extremadamente Complejo	3,5	4,25	2	3	2,4	3,03	2	4,25

	Indique el tiempo aproximado que suele tomar la fase formal de Codificación en un Proyecto de Desarrollo de Software, de acuerdo a su grado de complejidad. Por favor indique las unidades que toma como referencia (horas, dias, semanas, meses, etc).							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	1	2,5	0,75	0,5	0,75	1,1	0,5	2,5
Poco Complejo	2	4,5	1	0,75	1,25	1,9	0,75	4,5
Medio Complejo	3	10	1,5	1,2	1,7	3,48	1,2	10
Muy Complejo	5	12	2	2	3	4,8	2	12
Extremadamente Complejo	12	18	2,5	7,5	6	9,2	2,5	18

	Indique el tiempo aproximado que suele tomar la fase formal de Pruebas en un Proyecto de Desarrollo de Software, de acuerdo a su grado de complejidad. Por favor indique las unidades que toma como referencia (horas, dias, semanas, meses, etc).							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Sencillo	0,5	0,5	0,25	0,5	0,25	0,4	0,25	0,5
Poco Complejo	0,75	0,75	0,25	0,75	0,25	0,55	0,25	0,75
Medio Complejo	0,75	1	0,25	1	0,5	0,7	0,25	1
Muy Complejo	1	1,5	0,5	1,2	0,75	0,99	0,5	1,5
Extremadamente Complejo	1	2	0,5	3	0,75	1,45	0,5	3
	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después del Diseño necesita ser Re-Analizado, en cada caso:(Flujo A de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	20	20	40	10	20	26	10	40
Proyecto Poco Complejo	25	20	40	12	25	27,9	12	40
Proyecto Medio Complejo	25	40	60	16	40	40,95	16	60
Proyecto Muy Complejo	30	40	60	18	40	42,1	18	60
Proyecto Extremadamente Complejo	40	60	80	20	50	55,5	20	80
	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después la Codificación necesita ser Re-Analizado, en cada caso:(Flujo B de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	10	20	10	50	10	19	10	50
Proyecto Poco Complejo	15	40	15	60	12	26,05	12	60
Proyecto Medio Complejo	15	40	15	70	15	28,5	15	70
Proyecto Muy Complejo	15	60	30	80	25	40	15	80
Proyecto Extremadamente Complejo	20	80	30	90	35	46,25	20	90
	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después la Realización de las pruebas necesita ser Re-Analizado, en cada caso:(Flujo C de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	10	20	15	10	10	13	10	20
Proyecto Poco Complejo	10	40	15	15	10	16	10	40
Proyecto Medio Complejo	10	40	30	20	15	23,75	10	40
Proyecto Muy Complejo	15	80	40	35	15	35,5	15	80
Proyecto Extremadamente Complejo	15	100	40	50	20	41,25	15	100

	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después la Codificación necesita ser Re-Diseñado, en cada caso:(Flujo D de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	10	20	10	10	15	11,75	10	20
Proyecto Poco Complejo	10	20	15	15	15	14,75	10	20
Proyecto Medio Complejo	15	40	15	20	20	19,25	15	40
Proyecto Muy Complejo	20	60	30	25	20	29	20	60
Proyecto Extremadamente Complejo	20	80	30	30	30	33,5	20	80
	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después la Realización de las pruebas necesita ser Re-Diseñado, en cada caso:(Flujo E de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	10	20	10	10	5	10,25	5	20
Proyecto Poco Complejo	10	40	10	20	10	15	10	40
Proyecto Medio Complejo	15	60	15	30	15	22,5	15	60
Proyecto Muy Complejo	15	80	15	40	20	27,25	15	80
Proyecto Extremadamente Complejo	20	100	30	50	25	38,75	20	100
	Indique el porcentaje del flujo de trabajo que, bajo su apreciación, después la Realización de las pruebas necesita ser Re-Codificado, en cada caso:(Flujo F de la imagen de arriba)							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Media**	**Mini mo**	**Máxi mo**
Proyecto Sencillo	10	20	5	5	5	7,25	5	20
Proyecto Poco Complejo	15	40	5	8	10	11,35	5	40
Proyecto Medio Complejo	15	60	10	10	15	16,5	10	60
Proyecto Muy Complejo	25	60	15	15	20	21,75	15	60
Proyecto Extremadamente Complejo	30	80	25	20	25	30,25	20	80
	Basado en su experiencia, ¿Se suele incorporar la creación de un glosario, como una tarea extra de sus proyectos de software?:							
	Enc. 1	**Enc. 2**	**Enc. 3**	**Enc. 4**	**Enc. 5**	**Siemp re**	**50/50**	**Otros**
	Siemp re	Siemp re	Siemp re	50/50	Siemp re	4 de 5	1 de 5	0
	¿Cómo considera es su experiencia y/o familiarización con los asuntos que se mencionan en este cuestionario?							
	Enc.	**Enc.**	**Enc.**	**Enc.**	**Enc.**	**Alta**	**Exper**	

	1	2	3	4	5		to	
	Alta	Alta	Exper to	Exper to	Alta	3 de 5	2 de 5	