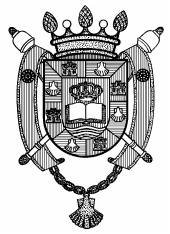
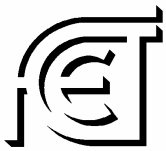


UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS



LICENCIATURA EN SISTEMAS DE INFORMACIÓN

TRABAJO FINAL DE GRADUACIÓN

MODELO DINÁMICO EXTENDIDO PARA LA GESTIÓN DE PROYECTOS SOFTWARE

Autor:

JUDITH ALEJANDRA LASSALLE

Profesor Guía**:**

MABEL SOSA

Noviembre de 2009

UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍAS

TRABAJO FINAL DE GRADUACIÓN DE LA LICENCIATURA EN SISTEMAS DE INFORMACIÓN

MODELO DINÁMICO EXTENDIDO PARA LA GESTIÓN DE PROYECTOS SOFTWARE

Autor:

.....
Judith Alejandra Lassalle

Profesor Guía:

.....
Mabel Sosa de Goldar

∗ ---------- ∗ ---------- ∗

Aprobado el día ..... del mes de ..... del año 20.....
por el Tribunal integrado por

.....	.....	.....
(firma)	(firma)	(firma)
.....	.....	.....
(aclaración)	(aclaración)	(aclaración)

Santiago del Estero – Argentina

Dedicatorias:

A mis padres, por la vida y el amor incondicional,
por ser el mejor modelo de esfuerzo que me permitió alcanzar este logro.
A mi abuela Mabel y a la memoria de mis abuelos
Alcira, Ramón y Osmar, por la presencia permanente en este largo camino cuesta arriba.
A mis hermanas Claudia y Mónica por su preocupación y palabras de aliento.
A mis sobrinos Aminna, Ignacio y Franco por la alegría y los momentos de juegos y risas que hicieron cable a tierra mis tensiones.

Agradecimientos:

Al Ing. En Sistemas César Eduardo González Pérez, miembro del grupo de investigación SIMON de la Universidad Industrial de Santander de Colombia, por su colaboración desinteresada en la conducción del razonamiento para la definición de las ecuaciones claves del modelo matemático propuesto.

A la Msc. Ing. Mabel Sosa por su guía y apoyo en la investigación.

A la Lic. En Sistemas de Información Mónica Daniela Cáceres Delazzari por su contribución en la etapa final de este trabajo.

ÍNDICE

ÍNDICE

RESUMEN

Resumen

Un proceso de desarrollo de software global posee numerosos desafíos y dificultades, así como también potenciales beneficios significativos. Para lograr un proyecto exitoso, las organizaciones del software necesitan adaptar y mejorar sus procesos para que soporten este tipo de desarrollo. Es aquí donde se requiere un sólido proceso de gestión de proyectos. Sin embargo hasta el día de hoy los proyectos de software siguen encontrando dificultades para ser entregados en tiempo, dentro del presupuesto y que satisfagan las necesidades de los clientes.

Estudios indican que las causas de estas fallas están relacionadas a problemas de la Ingeniería de Requisitos tales como el arrastre de errores de requisitos a fases posteriores, requisitos pobremente definidos, requisitos que son imposibles de satisfacer, requisitos que fallan en satisfacer las necesidades de los usuarios, surgimiento de nuevos requisitos, cambios de requisitos. Una adecuada gestión de requisitos puede formar la base para la estimación, planificación, seguimiento y control del progreso del proyecto y así reducir la posibilidad de fallas.

Reconociendo las bondades de los modelos de simulación para guiar el proceso de gestión de proyectos software, el presente trabajo formaliza un *Modelo Dinámico Extendido para la Gestión de Proyectos Software*, que toma como base un modelo dinámico, el de Abdel-Hamid y Madnick [1] y, mediante la adición de un módulo de la etapa de análisis omitida en dicho modelo, puede aplicarse integralmente a cada una de las etapas básicas del proceso de desarrollo y dar soporte a la toma de decisiones por parte de los directores del proyecto.

Para el estudio el modelo integrado, que se desarrolló bajo las convenciones de la Dinámica de Sistemas, se implementó como una herramienta de simulación bajo el entorno de la aplicación Evolution con la que se condujo la simulación del caso de estudio “Dynamics Explorer-Attitude”, un proyecto típico y de mediana envergadura. Para simular el proyecto, el *Modelo* se parametrizó estableciendo el valor para 21 parámetros relacionados con recursos humanos, desarrollo del software, planificación y estimación inicial para el tamaño, esfuerzo y duración del proyecto.

Con los parámetros definidos se ejecutó el *Modelo* extendido y se analizó su comportamiento comparando las salidas obtenidas con valores reales registrados para proyecto y valores obtenidos con el modelo base. En relación a la hipótesis planteada se

examinó la dinámica del comportamiento para las variables de tiempo, esfuerzo/costo, calidad, productividad. En relación a las variables *Duración* y *Esfuerzo* el experimento con el modelo extendido arrojó valores más próximos a los registros reales. En referencia a la variable *Calidad*, el resultado de la simulación demostró la bondad del ciclo de gestión de requisitos incluido en modelo extendido propuesto, ya que se obtuvo como salida una considerable reducción del número de errores. Por otra parte, en relación a la variable *Productividad* los resultados mostraron que al cabo del mismo tiempo se completa un mayor porcentaje de tareas utilizando el *Modelo* extendido.

Los resultados de la simulación muestran que el *Modelo* integrado propuesto es capaz de proporcionar medidas que permiten cuantificar con mayor precisión en las tareas de gestión de un proyecto de desarrollo de software.

Palabras claves: proceso de desarrollo de software, gestión de proyectos software, ingeniería de requisitos, gestión de requisitos, modelo de simulación.

INTRODUCCIÓN

.....

Introducción

Desde los primeros desarrollos de Sistemas de Información (SI) hasta los actuales, un problema fundamental ha sido el cumplimiento de los plazos de entrega dentro de los costes establecidos, así como poder realizar un seguimiento y control de la evolución de los proyectos, por lo que el establecimiento de métodos que permitiesen alcanzar estos objetivos de una forma lo más realista y exacta posible ha sido un factor cada vez más importante para la Ingeniería del Software.

Sin embargo, a pesar de que la Ingeniería del Software ha conseguido notables éxitos, también es cierto que ha caído en lo que se ha denominado la "crisis del software", término que ha sido utilizado desde el comienzo de esta disciplina y momento desde el cual, los ingenieros del software han estudiado gran variedad de metodologías, tecnologías y disciplinas, pero ninguna de las cuales permite resolver totalmente el problema de la crisis del software. Prueba de ello es que al día de hoy todavía sigue sin ser posible cuantificar con exactitud los plazos, costes, recursos humanos y técnicas que lleven a un desarrollo exitoso del software, tal y como otras ramas de la ingeniería en otros campos sí han sido capaces de hacer [27].

Esto refleja que no es suficiente avanzar sistemáticamente a través de etapas tradicionales de construcción de software y esperar un producto satisfactorio al final del mismo. El proceso de producción del software tiene que ser gestionado de una manera rigurosa y cuantitativa, de modo que se pueda verificar que el trabajo correspondiente a cada etapa se ha realizado dentro de los *plazos de tiempo* y *coste establecidos* y de acuerdo con *estándares específicos de calidad*. Un primer paso para dar lugar a esta gestión exitosa es mirar más allá de aspectos meramente técnicos (producción), asimilando la idea de que un SI funciona dentro de un contexto socio-cultural-político (aspectos humanos y organizacionales).

Afortunadamente existen muchos métodos/modelos creados para apoyar la gestión de los proyectos de desarrollo de software. Los más aceptados y utilizados son estáticos y tienen una base empírica [23], por lo que, bajo estas características, no sería asombroso que fallen, por dos razones: la primera porque, como todo proceso, un Proceso de Desarrollo de Software es de naturaleza dinámica, es un sistema dinámico, que cambia continuamente a lo largo de sus etapas [18]. La segunda, porque son aplicados en entornos de desarrollo diferentes a los que los originaron [24] [25] [26].

La perseverancia de la problemática en la gestión de proyectos software ha conducido a los implicados en la industria del software por numerosas investigaciones enfocadas en descubrir las causas y posibles soluciones más allá de los avances técnicos y metodológicos ya alcanzados. Comenzando por los primeros resultados de estudios de Siegel en 1990 [31], siguiendo con el informe "Chaos" realizado por The Standish Group en 1994 [32], e incluyendo experiencias prácticas compartidas en eventos internacionales recientes como "Solo Requisitos 2008" [4], que congregan profesionales del software que dirigen organizaciones mundialmente reconocidas (Softtek, Telelogic, Borland, entre otras) coinciden en que, por ser los requisitos los cimientos sobre los cuales se construye un producto software, de las buenas prácticas en la definición y gestión de requisitos depende que los productos se entreguen en tiempo, dentro de los costos y con la calidad esperada por los clientes.

Así, ante los persistentes fracasos registrados en las tareas de gestión de desarrollo de software (estimación, planificación, seguimiento y control) por contar con el apoyo de modelos solo "aparentemente" confiables y habiendo resaltado la relevancia de la Ingeniería de Requisitos (IR) como pieza clave de oportunidades para el éxito de un proyecto, se percibe la necesidad de desarrollar modelos dinámicos, los cuales asumen explícitamente que una multiplicidad de variables en relación, como esfuerzo y costes de un proyecto software, cambian a lo largo de las distintas etapas del proceso de desarrollo, considerando q cada una de ellas tiene una contribución de valor para el éxito no menos importante que las otras. Son estos modelos los que permiten realizar estimaciones y simulaciones que predicen los cambios en los costos, las necesidades de personal o el incumplimiento del calendario previsto [13] [25].

Habiendo destacado la importancia de la definición y gestión de los requisitos para una eficaz gestión de proyectos software y señalado las limitaciones de los modelos/métodos de gestión existentes, el presente trabajo formaliza un modelo dinámico extendido, que tomando como base el primer modelo dinámico, el de Abdel-Hamid y Madnick, reconocido como referencia obligada para desarrollos posteriores a él, lo extiende añadiendo a modo de sub-módulo, un modelo del proceso de la Ingeniería de Requisitos, ya que precisamente la carencia del modelo de Abdel-Hamid y Madnick es la omisión de la etapa de análisis.

Con este desarrollo se lograría un modelo dinámico que abarca integralmente el proceso de desarrollo de software desde el análisis hasta la prueba y la interacción entre

estas etapas, que potencie de manera efectiva el desempeño del proceso de gestión de proyectos software.

El trabajo de investigación desarrolla el modelo dinámico extendido según la Metodología de la Dinámica de Sistemas (DS) como herramienta para modelar y de simulación, que de soporte a la tarea de gestión de proyectos de desarrollo de software. Desde estas dos perspectivas de la DS (modelo y simulación) será posible:

- Visualizar las interacciones entre las diferentes variables que intervienen en el proceso de gestión y los cambios que en ellas se producen, a lo largo del proceso de desarrollo, en tiempos muy reducidos.
- Determinar las consecuencias de las relaciones multivariables a través del análisis de los resultados arrojados por la simulación, generados bajo diferentes condiciones.

Se prevé que, con el apoyo de la herramienta de simulación derivada de este modelo, el equipo de gestión podrá contemplar un amplio rango de situaciones *antes* de tomar las decisiones relevantes en la gestión del proceso de desarrollo de software, para posteriormente formular políticas de decisión efectivas para lograr el éxito del proyecto.

Este trabajo ser organiza de la siguiente forma. En el Capítulo I se enuncian los problemas y necesidades que enmarcan el estado de la cuestión y se definen los objetivos e hipótesis que forman el eje conductor del presente trabajo. Los marcos referenciales que sintetizan los conceptos y el contexto de la investigación, se detallan en el Capítulo II. Luego en el Capítulo III se describe el modelo tomado como base, el modelo de Abdel-Hamid y Madnick. El Capítulo IV detalla el modelo extendido derivado del modelo base con la descripción de un submodelo de la etapa de análisis y una interfase. Este capítulo también contiene una descripción de la herramienta de simulación derivada del modelo extendido. La prueba del modelo global con la experimentación de un caso de estudio bajo la herramienta de simulación, se muestra en el Capítulo V.

Finalmente, se presentan las Conclusiones del trabajo realizado, seguido de las Referencias Bibliográficas y los Anexos.

CAPITULO I

.....

PROBLEMAS. NECESIDADES. OBJETIVOS. HIPÓTESIS

I.1. *Problemas*

I.1.1. *Observación del Fenómeno*

Como cualquier otra organización abocada a la producción, las dedicadas al desarrollo de software mantienen entre sus principales fines, la producción de software de acuerdo con la planificación inicial realizada, además de una constante mejora con el fin de lograr los tres objetivos últimos de cualquier proceso de producción: alta calidad y bajo costo, en el mínimo tiempo. La gestión de un Proceso de Desarrollo de Software engloba todas las funciones que mantengan a un proyecto dentro de unos objetivos de costo, calidad y duración previamente estimados. La mayoría de estas funciones y técnicas de gestión y control empleadas, se han importado de otras industrias de producción que desarrollaron estos métodos a principios de siglo pasado [28].

Pero aún con la ayuda de estos primeros métodos no se ha podido superar aún la llamada "Crisis del Software", ya que la industria de desarrollo de software sigue caracterizándose por ejecutar proyectos para los que, con demasiada frecuencia, se han alcanzado límites elevados de imprecisión en relación con los niveles de plazo, costo y calidad inicialmente previstos.

Esto se debe al hecho que estos métodos tradicionales (COCOMO, SLIM, SofCost, PERT, GANT, etc.), tan arraigados en la cultura de la Ingeniería de Software, son *estáticos, empíricos y univariables* [26], que si bien son fáciles de usar y requieren poca información inicial, utilizan fórmulas derivadas empíricamente (obtenidas a través de una muestra reducida de proyectos), provienen de entornos de desarrollo muy concretos y organizaciones específicas y se basan en el tamaño del proyecto. Es por ello que son modelos de validez cuestionable en ambientes de desarrollo distintos a aquel en el que se obtuvieron los datos y sólo facilitan el punto de partida en la actividad de gestión, librando al azar la cuantificación y medición de las diferentes variables del resto de las etapas de desarrollo del software que sin dudas, por su naturaleza dinámica, estarán afectadas por un gran número de variaciones [21] [29].

Además, estas herramientas de estimación y seguimiento reportan una gran cantidad de *datos* que un director del proyecto debe ser capaz de interpretar y utilizar hábilmente en la consecución de sus objetivos. Sin embargo, dada la complejidad que encierran los proyectos de software por su propia naturaleza dinámica, es necesario

disponer de alguna herramienta o *modelo dinámico* que sea capaz de suministrar *información* (y no datos) acerca de la evolución en el tiempo de los proyectos en ejecución, a partir de la cual se puedan tomar las decisiones de gestión pertinentes [32]. Desde esta perspectiva, es preciso reconocer que la gestión no es una actividad meramente técnica a través de la cual se obtienen medidas de determinados atributos del proceso de desarrollo de software, sino que por el contrario, permite acceder y utilizar información derivada de un complejo dinamismo que caracteriza al entorno socio-técnico-político-cultural de la organización que desarrolla y/o utiliza el producto software.

Existen algunos modelos dinámicos, MDB (Modelo Dinámico Básico) [1] y MDI (Modelo Dinámico Intermedio) [27], que intentan apoyar la compleja tarea de la gestión del proceso de desarrollo de software entendida en sus aspectos socio-técnico-político-cultural, sin embargo éstos presentan la limitación de ser *parcializados*, ya que su aplicabilidad depende del nivel de conocimiento que se tenga del proyecto, conocimiento que a su vez depende, fundamentalmente, de la etapa de desarrollo en la que se encuentre el proyecto. Es decir que estos modelos se centran en etapas particulares del proceso de desarrollo. Por su parte el MDB sólo puede ser aplicable en las primeras etapas de desarrollo, cuando se tiene poca información sobre el proyecto a realizar y se necesita tener una primera aproximación de cuál podría ser la evolución de las variables fundamentales del mismo. Mientras que el MDI podrá ser aplicable en etapas más avanzadas, cuando existe conocimiento de un número importante de atributos del proyecto y de la organización de desarrollo.

El MDI sólo contempla las etapas de diseño, codificación y prueba, dejando de lado la etapa inicial del proceso de desarrollo que es el análisis y la última que es el mantenimiento del producto software. Por ser el MDB una reducción del MDI, posee las mismas carencias.

Es de esperar entonces, que del modelo que se aplique dependerá el nivel de detalle de la información que se obtenga para llevar a cabo las tareas de gestión, y puesto que estos modelos existentes no contemplan la totalidad del proceso de desarrollo, ya que lo limitan a etapas específicas (diseño, codificación y prueba), dejan de lado una gran cantidad de atributos del proyecto y de la organización de desarrollo involucrados en las otras etapas no menos importantes por las que va atravesando el software, que son necesarios conocer para una gestión global eficaz.

I.1.2. *Enunciado del Problema*

Las medidas de éxito para un responsable del proyecto son muy simples, lograr la satisfacción del cliente finalizando el proyecto a tiempo y dentro del presupuesto inicialmente previstos. Sin embargo, en la práctica, los problemas a los que se enfrenta el responsable de proyectos cuando intenta cumplir estas expectativas son complejos.

Esta complejidad radica en que los proyectos de desarrollo de software son altamente dinámicos en su naturaleza. Mientras un proyecto evoluciona y los eventos progresan, rara vez se ajustan al plan de trabajo documentado. Muchos factores contribuyen a que el proyecto se desarrolle a un ritmo diferente al del plan. Cuando el progreso percibido es diferente al anticipado, se hacen ajustes al proyecto en tiempo real para adaptarse a los cambios. Estos ciclos de realimentación inherentes deberían ser capturados en un modelo de proceso para que sea realista [17].

Pero si bien la Ingeniería de Software ha avanzado en el desarrollo de metodologías, herramientas y estándares que aseguren el éxito de los proyectos a través de modelos de procesos de creación de software más productivos y rentables, atendiendo a su perspectiva dinámica, poco se ha considerado en relación a las posibilidades de mejora del proceso desde un área de oportunidad declarada como las más prometedora: *la etapa de análisis de requisitos* [2] [9] [16] [18] [33].

Estadísticas [5] [12] [13] [25] [34] han revelado que más del 50% de las causas de fracasos en la gestión de proyectos están relacionadas a una inadecuada Ingeniería de Requisitos.

Más aún, todavía se ejecutan proyectos en los que la captura de requisitos sucede en forma *estática y cerrada*, sugiriendo que un proyecto es ordenado y progresa en etapas bien definidas y predecibles hasta su finalización, lo cual es más falso cuanto más grande y complejo es el sistema que se va a desarrollar [14].

Sin embargo, aunque se ha reconocido a la etapa de análisis de requisitos como área de proceso clave para el éxito de los proyectos el problema persiste y, en forma general, se puede plantear de la siguiente manera:

*El **proceso de gestión de proyectos software** carece de modelos de soporte **dinámicos e integrales** que incluyan la **etapa de análisis de requisitos** y su interacción con el resto de las etapas del proceso de desarrollo (diseño, codificación y prueba).*

"No Silver Bullet" [6] expone que todavía no hay una bala de plata que pueda acabar con el problema que impide que el software funcione correctamente en tiempo y en costo.

A pesar de los avances, desde la perspectiva de la gestión de proyectos de desarrollo de software, esta ley sigue siendo válida al día de hoy.
Cada año los costos del desarrollo de software continúan aumentando. Comparado a otras áreas de ingeniería, los esfuerzos de software experimentan números elevados de desarrollos que terminan con costos y tiempos desbordados [7].

A partir del enunciado del problema se pueden formular los siguientes interrogantes que guían el desarrollo del presente trabajo:

- ¿Es posible introducir **mejoras en el proceso de gestión de proyectos software** mediante la incorporación de modelos dinámicos del proceso de Ingeniería de Requisitos a los modelos de soporte a la gestión?
- ¿La formalización de un modelo dinámico de gestión, centrado en la mejora de la etapa de análisis, puede incrementar la predictibilidad de **costos** y **tiempos** del proyecto software?
- ¿La formalización de un modelo dinámico de gestión, centrado en la mejora de la etapa de análisis, puede incrementar la **calidad** del producto software?
- ¿La incorporación de un modelo del proceso de Ingeniería de Requisitos a los modelos de soporte de la gestión, puede aumentar la **productividad** en el proceso de desarrollo de software?

I.2. *Necesidades*

Mohanty [23], en un estudio de aplicación de diferentes modelos de estimación empírica (tradicionales), resalta la importancia que tienen en los métodos de estimación no sólo los aspectos técnicos del entorno de desarrollo sino también las diferentes políticas de gestión que se aplican. Estas políticas dependen de las características propias de la organización e inclusive de los propios directores de proyectos, aspectos que la mayoría de estos modelos no consideran.

Atendiendo a estas necesidades, a principios de la década de los ’90 se presenta el primer modelo dinámico de Abdel-Hamid y Madnick, el cual incorpora el entorno socio/político de la organización, su nivel de madurez y todas las actividades relacionadas con la gestión de recursos humanos, producción de software, planificación y control.

Este modelo y los derivados de él, están basados en la Dinámica de Sistemas cuya utilidad, en el contexto de la gestión, radica en distintos aspectos:

a).- Entender las interacciones entre las variables relevantes que traman la complejidad dinámica de la gestión de un proyecto,

b).- Formular políticas de decisión efectivas para lograr el éxito del proyecto,

c).- Conocer los efectos secundarios de las decisiones tomadas sobre la globalidad del proyecto.

Esto es posible ya que los proyectos pueden ser simulados sin el riesgo de utilizar recursos de manera inadecuada y además con el beneficio de adquirir el aprendizaje necesario para aplicarlo posteriormente en la fase de inversión o ejecución.

Por supuesto, no se puede negar que los trabajos realizados en el contexto de los modelos dinámicos de soporte a la gestión son importantes, pero aún así, existe una importante brecha por superar. Ésta reside en la *parcialidad* del primer modelo dinámico, de Abdel-Hamid y Madnick, y por consiguiente de los otros modelos que de él se derivan, por no tratar integralmente todas las etapas del proceso de desarrollo de software, ya que sólo contemplan las fases de diseño, codificación y prueba. Esto lleva a que en el proceso de gestión no se contemplen aquellos atributos que intervienen en la primera y tan importante etapa como lo es la de *análisis* y que sin duda influyen en el resto de las etapas del proceso de desarrollo.

Precisamente, la atención de investigaciones de los implicados en la industria del desarrollo de software orientada hacia el campo de la gestión de proyectos, llevó a indicar

que las causas mayores de fracasos están en áreas como el ambiente social y político, los acuerdos legales y los factores humanos [14]. Por ser estas áreas las que caracterizan la etapa de análisis de requisitos, se puede indicar que esta etapa del proceso de desarrollo de software es un área crítica de oportunidad para la mejora del proceso de gestión en la consecución de proyectos exitosos.

Es por ello que el presente trabajo, partiendo de las ventajas de la metodología Dinámica de Sistemas y de la carencia de modelos dinámicos globales de gestión, busca llenar ese vacío con la formalización de un ***modelo dinámico extendido para la gestión de proyectos de desarrollo de software*** que, tomando como base el primer modelo dinámico (de Abdel-Hamid y Madnick) y mediante la adición de la etapa de análisis omitida en dicho modelo, pueda aplicarse integralmente a cada una de las etapas básicas del proceso de desarrollo de software (análisis, diseño, codificación y prueba). Así, tratando las particularidades de cada una de ellas, las cuales aportan diferentes niveles de conocimiento sobre el proyecto y la organización, y al mismo tiempo acoplando los resultados obtenidos en cada etapa se puede analizar la evolución global del proyecto, apoyando así a la tarea de toma de decisiones en la gestión del proceso de desarrollo.

La utilidad de este modelo sería doble: por un lado, permitiría analizar de manera aislada las singularidades de las etapas de análisis, diseño, codificación y prueba en relación a la gestión y, por otro, permitiría enlazar los resultados de una etapa con la siguiente para analizar las repercusiones que tiene una etapa concreta sobre el resto del proyecto. Esto permitirá disponer de información más confiable para tomar decisiones que conduzcan a resultados de gestión exitosos.

Al mismo tiempo, con este modelo, el aporte al área de gestión de proyectos de desarrollo de software (Ver Figura I.1), sería doble:

- *Teórico:* consistiría en la definición conceptual y matemática del modelo propuesto correspondiente a la *fase de análisis* y su *interfaz* con las etapas restantes del proceso de desarrollo de software (diseño, codificación y prueba) contempladas en el modelo tomado como base, obteniendo así un modelo integrado que abarque las etapas análisis, diseño, codificación y prueba y contemple además el enlace entre ellas, para conseguir una gestión global del proceso.

- *Práctico:* consistirá en la construcción de una herramienta de simulación, derivada de la formalización matemática que se indica en el aporte teórico, la cual permitirá

ensayar diferentes alternativas de gestión constituyendo de esta manera una guía para el director del proyecto en la toma de decisiones que conduzcan al éxito del mismo.

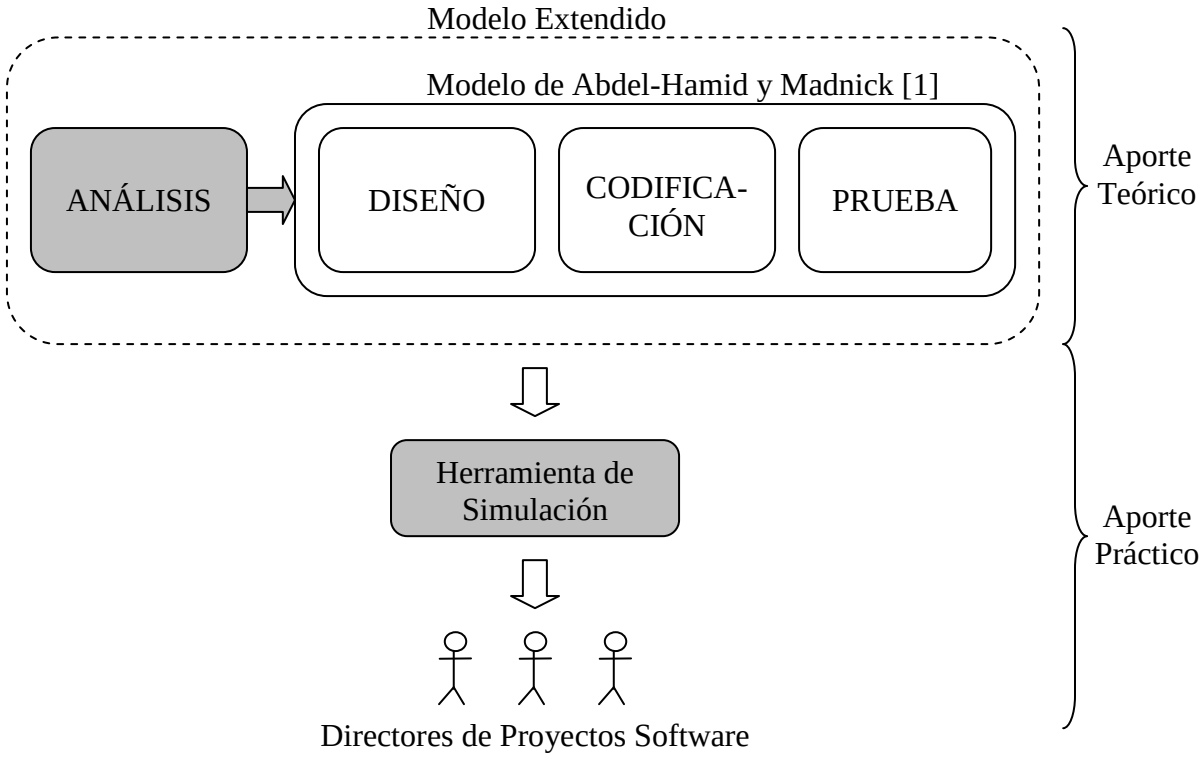


Figura I.1. Esquema del Trabajo Propuesto

En la Figura I.1. se muestran las actividades implicadas en el presente trabajo de investigación. La originalidad del trabajo (indicada en recuadros sombreados) involucra, por un lado, la definición del modelo correspondiente a la etapa de *análisis* y su *interfaz* con las etapas restantes del proceso software (diseño, codificación y prueba) contempladas en el modelo base de Abdel-Hamid y Madnick. Por otro lado, a partir del modelo integrado, se deriva una herramienta de simulación cuyos resultados servirán de guía a los directores de proyectos en el proceso de toma de decisiones relacionadas con la gestión de proyectos de desarrollo de software, que abarque desde la etapa de análisis hasta la etapa de prueba.

I.2.1. *Necesidad de un Modelo Dinámico Integral*

I.2.1.1. *¿Por qué desarrollar un modelo dinámico de simulación para la Gestión de Proyectos Software?*

Un ***modelo de simulación*** es un modelo computacional que consiste en la abstracción o representación simplificada de un sistema dinámico complejo. Estos modelos tienen la ventaja de poder experimentar diferentes decisiones y analizar sus resultados en situaciones donde el costo o el riesgo de una experimentación real son excesivos. Además, los modelos de simulación permiten el estudio de sistemas complejos que resultan difíciles de representar mediante modelos estáticos [23].

Lo que da origen a dicha complejidad es *la incertidumbre, el comportamiento dinámico y la retroalimentación* que caracteriza a la mayoría de los sistemas reales.

En un proyecto de desarrollo de software (PDS), no es excepcional encontrar estas características. Es sabido que existen aspectos o áreas de un PDS y de la organización que lo lleva a cabo, sobre los que no se tiene conocimiento o este es muy vago (*incertidumbre*); también, las variables que participan en dicho proceso toman diferentes valores conforme se avanza en el ciclo de vida (*comportamiento dinámico*), y a su vez cambios en esas variables influyen sobre la evolución global del proceso (*retroalimentación*).

En [22], para destacar las características dinámicas de un proceso de desarrollo de software, se expresa que éste es un sistema dinámico complejo y cuya complejidad promueve un comportamiento que no puede ser evaluado precisamente sólo a la luz de la experiencia humana.

Un proceso de desarrollo de software presenta las siguientes características:

- *Varios componentes interrelacionados:* un sistema software esta formado por varios componentes interrelacionados. Las relaciones entre estos elementos son complejas y pueden causar un comportamiento no intuitivo. Este indica que hay implicaciones de un cambio en un componente del software que requiere análisis más profundos para comprender sus impactos.
- *Comportamiento dinámico complejo:* el desarrollo de software es una tarea dinámica. Muchos elementos del proyecto, tales como el equipo de desarrollo, partes producidas, varían a lo largo del proceso de desarrollo. Los efectos de una

acción aplicada a un proceso de desarrollo, se perciben mucho más tarde que la acción misma. Así, el proceso de desarrollo de software puede presentar relaciones causa-efecto que pueden ser distantes en el tiempo, lo cual es una importante característica dinámica.

- *Bucles de realimentación:* los procesos de desarrollo de software presentan bucles de realimentación, donde los efectos de una acción refuerzan las condiciones que promovieron su aplicación.
- *Relaciones no lineales:* los proyectos de desarrollo de software presentan muchas relaciones no lineales entre sus componentes. Estas relaciones son identificadas cuando los efectos de una acción no son proporcionales a los de la acción original.
- *Manipulación de datos blandos:* un proyecto de desarrollo de software no es meramente un tema de ingeniería, es esencialmente un esfuerzo humano y no puede ser considerado solamente en términos de sus componentes y sus relaciones. Es por ello que cierta información cualitativa, tal como la motivación del equipo, el cansancio de los desarrolladores, las características específicas de la organización, son relevantes para determinar los atributos del proyecto, como su costo y tiempo de conclusión.

A partir de las características mencionadas anteriormente, se infiere la necesidad de contar con un modelo dinámico de simulación de un proceso software. Este modelo proporcionaría los medios a través de los cuales se pueda experimentar para obtener conocimiento del propio proceso de desarrollo y de la organización para predecir su comportamiento/evolución y su influencia sobre el resto del proceso y la organización.

Existen otras razones para desarrollar un modelo de simulación del proceso software miradas desde la perspectiva de la gestión. En la mayoría de las ocasiones la simulación se comporta como una ayuda en el proceso de toma de decisiones, también favorece el estudio y disminución de los riesgos y asesora a la dirección en los niveles estratégico, táctico y operacional.

Dentro de esas razones, las principales son [22]:

- ***Gestión estratégica***: La simulación puede ayudar a resolver un amplio rango de cuestiones relacionadas con la gestión estratégica. En estos casos, los modelos de simulación recogen el comportamiento de la organización en una serie de

parámetros que asumirán valores diferentes según la cuestión que se trate de resolver. Los directores de proyectos pueden comparar los resultados de los diferentes escenarios simulados obteniendo un conocimiento extra que les ayude en la toma de decisiones.

- ***Planificación***: La simulación del proceso software se puede aplicar tanto en la planificación inicial del proyecto como en las sucesivas planificaciones o modificaciones que se realicen conforme el proyecto progresa. La predicción del esfuerzo/costo, el tiempo de desarrollo, la calidad del producto, los niveles de personal necesarios, etc. son algunas de las posibilidades que se pueden estudiar mediante la simulación de diferentes escenarios.
- ***Control y gestión operacional***: La simulación también puede proporcionar un soporte efectivo para las actividades de control y gestión operacional de los proyectos software. La evolución de las variables claves del proyecto (como el estado actual del progreso, el consumo de recursos, la calidad percibida, etc.) puede ser monitoreada y comparada con los valores planificados. En estos casos resultará imprescindible disponer de información detallada, actualizada y precisa del proyecto cuyo modelo se simula.
- ***Mejora del proceso y cambio tecnológico****:* La simulación puede facilitar el proceso de toma de decisiones en el ámbito de la mejora del proceso ya que permite predecir el impacto potencial de un cambio en el proceso antes de que éste se haga efectivo en la práctica. Como consecuencia, también será posible utilizar la simulación en el diseño de procesos estándares específicos para una determinada organización. En este mismo sentido, la simulación también puede facilitar la toma de decisiones relacionadas con un cambio de la tecnología empleada dentro de la organización.
- ***Comprensión***: El propio proceso de construcción de un modelo de simulación permite ampliar el conocimiento que se tiene acerca de la organización y de sus procesos de desarrollo, ya que resulta necesario conocer y comprender correctamente los lazos de realimentación, sus efectos y los retrasos existentes en el proceso para que el modelo sea preciso. Además, en esta etapa de construcción del modelo es probable que se identifiquen aquellas áreas del proceso donde la incertidumbre es mayor y resulte más complicado, al igual que en la realidad, predecir la salida. Finalmente, los modelos de simulación

favorecen la comunicación entre los miembros de la organización ya que permiten compartir, bajo una representación formalizada, el conocimiento que cada miembro tenga del proceso software.

- ***Formación y aprendizaje**:* No es una novedad que la simulación se defina como una herramienta eficaz para la formación y el aprendizaje. La simulación es un medio para que el personal pueda practicar o aprender gestión de proyectos. Un entorno de entrenamiento basado en simulación permite aprender los resultados más probables de las decisiones de gestión más frecuentes y que, a menudo, resultan incorrectas como, por ejemplo, adelantar excesivamente la etapa de codificación, eliminar las revisiones o reducir el esfuerzo asignado a las actividades de prueba. El entrenamiento basado en la simulación también ayuda a los directores de proyectos a aceptar resultados muy diferentes a los que esperaban cuando tomaron la decisión. De esta manera, se consigue que el director obtenga una experiencia que, de otra forma, sólo hubiera adquirido tras años de experiencia gestionando proyectos reales.

Por lo expuesto precedentemente, un modelo dinámico de simulación es una potente herramienta para la ***gestión de proyectos***.

Si la gestión de un proyecto software se considera compuesta por una serie de actividades que se encuentran integradas, en el sentido de que una determinada acción desarrollada en un área afectará a otras áreas, un modelo dinámico de simulación favorece la comprensión de la naturaleza integrada de la gestión de proyectos al describirla a través de sus procesos, estructuras e interrelaciones principales. Por ejemplo, a través de un modelo tal, se puede analizar como un retraso en el calendario afectará al costo del proyecto, y si afecta o no a la motivación del equipo o a la calidad del producto. O bien, analizar cómo la acción de reducir la calidad o los requisitos implementados en una versión concreta del producto software contribuye a cumplir las previsiones temporales o de costo.

Otra ventaja de los modelos de simulación que se suma al soporte de la gestión de los procesos de desarrollo de software, es que pueden emplearse cualquiera sea el nivel de madurez de la organización (según CMM - Capability Maturity Model) que los lleve a cabo, a la vez que les permite obtener conocimiento que les ayuda a avanzar en la obtención de una madurez mayor [23].

En el *nivel de madurez 1* se puede introducir la idea del proceso de software como una entidad dinámica cuyo comportamiento está gobernado por lazos de realimentación. De esta manera, el director del proyecto tomará contacto con los entornos de simulación y con la potencialidad y ventajas del empleo de este tipo de modelos.

Las organizaciones que pertenecen al *nivel 2* pueden comenzar a diseñar modelos de sus procesos y examinar algunas de las propiedades de sus comportamientos. En concreto, se pueden desarrollar modelos de gestión de proyectos muy generales (sin un alto nivel de detalle) que permitan simular aspectos tales como la planificación, el seguimiento y supervisión del proyecto. En este nivel de madurez, sólo se dispondrá de medidas muy generales (basadas en la experiencia en la mayoría de las ocasiones) relativas a los costos y al calendario. Las métricas del producto, como por ejemplo la densidad de errores, no están disponibles todavía. Por tanto, estos modelos de simulación sólo serán aproximados en cuanto al nivel de detalle con que se pueden construir y a la precisión de los datos que reciben en su entrada. No obstante, representan un comienzo importante para la predicción cuantitativa.

Las herramientas de simulación actuales permiten la rápida construcción de modelos muy sofisticados. Sin embargo hay que tener en cuenta que dadas las limitaciones de los datos métricos disponibles en el nivel 2, la validación de los modelos a este nivel es muy difícil de alcanzar. Por tanto, en este nivel se pueden utilizar los modelos para obtener avances cualitativos, pero su capacidad de predicción cuantitativa debe estar todavía cuestionada.

Ya en un *nivel 3* del modelo CMM, las organizaciones aplican un énfasis especial sobre la ingeniería del producto, la definición formal de los procesos de ingeniería y la instrumentación de estos procesos. Por tanto, la gestión (que inicialmente trataba a las actividades de ingeniería como cajas negras) posee ahora conocimiento sobre el interior de estas actividades. Los datos recogidos de la observación de los procesos de ingeniería pueden servir de soporte para las actividades de simulación, para comprender el comportamiento, estabilidad y sensibilidad de los cambios. Ya que los procesos de ingeniería conducen muchos de los procesos de gestión, la simulación precisa del nivel de ingeniería resultará en una precisión mejorada en el nivel de gestión. Por ejemplo, las tasas de corrección de errores afectan al tiempo de finalización de los módulos software y esto, tiene un efecto directo sobre el calendario del proyecto.

El nivel 3 también otorga una gran importancia a la definición, mantenimiento y reutilización de procesos en una organización. Esto implica que las organizaciones del nivel 3 deberían soportar una librería de los procesos que pueden reutilizarse, con una adaptación adecuada, en otras partes de la organización. Manteniendo las definiciones de los procesos como modelos de simulación, un usuario futuro de un proceso existente puede evaluar las características de evolución de los procesos dentro del contexto de un proyecto para un nuevo usuario y puede adaptarlo de una manera mucho más precisa que los procesos estáticos tradicionales.

Finalmente, en el nivel 3 la formación recibe un gran énfasis. En muchas industrias, la simulación es un componente esencial de esta actividad. Sin embargo, la industria del software no ha explotado aún esta aplicación. Como herramienta de entrenamiento, la simulación puede ayudar a mejorar el proceso de toma de decisiones. Las áreas claves de proceso como el seguimiento y supervisión, aseguramiento de la calidad e ingeniería del producto software son candidatos perfectos para el entrenamiento basado en simulación.

El énfasis de las prácticas de ingeniería en el nivel 3, hace que sea muy importante recoger métricas del producto. Estas métricas están asociadas con parámetros como la duración de las tareas, la estabilidad de los requisitos y los defectos de diseño y codificación y permitirán validar los modelos de simulación con un alto grado de confidencialidad. Además, las métricas de cada parámetro se deben recoger con el detalle suficiente para permitir la generación de distribuciones estadísticas. Las simulaciones estocásticas que utilizan estas distribuciones permitirán acceder a la incertidumbre de las variables independientes como, por ejemplo, la productividad del equipo técnico. Tener el conocimiento de la incertidumbre asociada con el costo o calendario planificado inicialmente sería de un valor apreciable en la gestión de riesgos y mejora del proceso.

En el progreso hacia el ***nivel 4***, la validación de los procesos simulados gana en rigor y refleja el nivel de confidencialidad ganado a través de la experiencia. Si se realizan modificaciones a los modelos del nivel 4 existe una alta probabilidad de que los cambios de comportamiento que exhiba el modelo se produzcan también en la realidad. Esto proporciona a la dirección la confianza de que las predicciones resultantes de la simulación son generalmente mejores que las que se basan exclusivamente en el juicio humano. Los directores son más capaces de asumir los riesgos de modificar sus procesos o integrar nuevos elementos en los ya existentes.

En el nivel 4, el objetivo es operar los procesos dentro de límites de rendimiento cuantitativo. La simulación es un medio para determinar cuáles deben ser esos límites. En primer lugar se deben determinar los límites de las variables dependientes (límites principales de costo, plazo y calidad). Estos límites actúan como restricciones de las variables independientes tales como densidad de errores, recursos, etc. A través de la simulación es posible determinar los límites superiores e inferiores que estas variables dependientes no deben exceder con el objeto de que el costo, la calidad y el plazo se mantengan dentro de los límites aceptables.

En este nivel, la simulación también va a permitir predecir (dentro de un intervalo de confianza) si el proyecto va a cumplir con éxito los objetivos del proyecto. Por otro lado, estas averiguaciones se pueden hacer en cualquier momento del ciclo de vida, basta con inicializar el modelo con una instantánea de los datos actuales del proyecto real.

A partir de las experiencias del nivel 4, se obtienen un conjunto de métricas de calidad, modelos validados que pueden utilizarse para seguir y dirigir (en tiempo real) el curso del proceso. En el *nivel 5* esta experiencia se utiliza para realizar cambios más radicales en los procesos, es decir, para cambiar los componentes principales de los mismos. En este nivel 5, se prueban mejoras o nuevas maneras de construir el software en un ambiente controlado. Insertar un nuevo componente en un proceso posee un elemento de riesgo independientemente de cuáles sean las técnicas de análisis. Pero las ventajas ofrecidas por el uso de la simulación ayudan a reducir el riesgo de manera significativa. Además, es posible comparar la salida de la simulación del cambio en un proceso, con la salida real de ese proceso sin modificar.

Un factor muy importante en el nivel 5 es la capacidad de responder rápidamente a la nueva tecnología, por ejemplo, las herramientas CASE y las herramientas de flujo de trabajo. La inserción de estas nuevas tecnologías posee importantes efectos, entre los que destacan los factores de riesgo humano, que la simulación no puede tener en cuenta de manera completa, pero su utilización puede orientar sobre sus efectos. Existe también una importante conexión entre la simulación, el flujo de trabajo y las métricas. Por un lado, la simulación puede identificar dónde puede instrumentarse un proceso (es decir, los puntos en los que los datos deben insertarse para conducir la simulación). Por otro lado, el flujo de trabajo nos ofrece la oportunidad de recoger métricas automáticamente en una rutina, de una manera precisa y no intrusiva. Por tanto, la simulación soporta el flujo de trabajo apuntando al lugar donde éste debe ser instrumentado, mientras que el flujo de trabajo proporciona los datos que permiten validar la simulación.

En síntesis, las organizaciones de nivel 5 poseen modelos detallados y validados para sus procesos. Son capaces de realizar cambios importantes en sus procesos, validarlos a través de la simulación y poseen un alto nivel de confianza que, cuando se lleva a la práctica hace que los procesos respondan de la misma manera que lo hace la evolución real. El énfasis sobre la inserción tecnológica que se realiza en el nivel 5, puede verse favorecido por la simulación que permite averiguar el impacto de los cambios de herramientas sobre el proceso antes de llevarlo realmente a la práctica.

Resumiendo, el desarrollo de software es un proceso complejo y dinámico ya que en él hay muchos factores interactuando a lo largo del ciclo de vida, que impactan sobre el costo y tiempo del proyecto de desarrollo y sobre la calidad del producto software desarrollado. Además, la industria del software constantemente se enfrenta a demandas crecientes de calidad, productividad, lo que hace de la ***gestión de proyectos de desarrollo de software*** una de las más difíciles y desafiantes tareas en cualquier organización de software. Por esto, no sorprende que la gestión de proyectos sea una de las áreas en las que las ***técnicas de simulación de procesos*** se hayan aplicado en el dominio de la ingeniería del software durante la última década, comenzando con el trabajo pionero de Abdel-Hamid T. K. y Madnick, S. E.

Con la publicación en 1991 de este primer modelo dinámico para la simulación de la gestión de proyectos de desarrollo de software, un significativo avance toma lugar en el campo de la gestión de proyectos. Se dio lugar a un área nueva la cual permitió una mejor comprensión de las diferentes variables a ser consideradas y las complejas relaciones entre ellas y la investigación del impacto de un cambio tecnológico, de diferentes políticas de gestión, del nivel de madurez de una organización sobre el proyecto en su totalidad antes de comenzar el desarrollo.

I.2.1.2. *¿Por qué agregar un módulo que contemple la etapa de Análisis en un modelo de Gestión de Proyectos Software?*

En palabras de Brooks [6]: “La parte más difícil en la construcción de sistemas software es decidir precisamente *qué* construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas software. Ninguna otra parte del trabajo

puede perjudicar tanto el resultado final si ésta se realiza en forma incorrecta. Ninguna otra parte es tan difícil de rectificar posteriormente". Ese *qué* construir se define en la primera etapa del desarrollo del software que es la etapa de Análisis.

Es esta etapa el punto de partida obligado de todo proyecto de desarrollo. Por lo tanto, el desarrollo del software sólo puede ser iniciado cuando se tiene bien establecido lo que se quiere producir. Si los desarrolladores no conocen de forma precisa el problema a resolver, no es probable que se obtenga una solución correcta y útil. El riesgo es obtener una solución técnicamente perfecta pero que no tenga utilidad.

Cuando se trata de sistemas complejos, esta definición de lo que se quiere no es trivial. Por ello, muchos productos de software no se comportan como sería deseable. Hacer una definición que cubra todas las necesidades de un sistema complejo es una tarea difícil.

Pero... ¿porqué tanta dificultad? Es que, como lo expresa Brooks, la definición de los requisitos es una etapa donde inevitablemente existe ambigüedad, incompletitud, contradicciones que atentan contra el correcto comienzo de la vida del producto. La definición de requisitos de un sistema complejo encierra volatilidad, incertidumbre, que se derivan de la participación de muchos usuarios (por lo que los requisitos entran en conflicto), de un entorno de negocios y técnico que cambia (nuevo hardware, nuevos sistemas, cambios en las prioridades de negocios, cambios legislativos, etc.) y de la propia maduración a causa del conocimiento adicional, fruto del desarrollo o de presiones del entorno o de la organización que no son previstas.

Por lo expresado anteriormente es que no resulta asombroso el hecho que los investigadores marquen como origen de la Crisis del Software las primeras etapas del desarrollo, y más, que las deficiencias en la definición y gestión de requisitos constituyan una de las principales fuentes del problema. Es un hecho comprobado, que los errores originados en la fase de captura de requisitos pueden permanecer sin ser detectados hasta la fase en la cual el sistema está implementado y cuando esto ocurre, según dichos estudios, el costo de un cambio de requisitos una vez entregado el producto es entre 60 y 100 veces superior al costo que hubiera representado el mismo cambio durante las fases iniciales del desarrollo, puesto que la propagación hacia fases posteriores de la construcción, provocan que los errores sean más graves y difíciles de corregir, requiriendo esto mayor esfuerzo, tiempo, y por lo tanto mayor costo.

Coincidentemente, otros trabajos de investigación, llevados a cabo por reconocidas instituciones del área (*"Software Engineering Institute"* - SEI y *"The Standish Group"*) encontraron que los constantes cambios a los requisitos y la incapacidad de gestionar estos cambios es una de las principales causas de que un producto software se entregue fuera de tiempo, exceda en costo y no cumpla con la calidad esperada por el cliente.

Como resultado de estos numerosos esfuerzos destinados a determinar las causas de la Crisis del Software, queda claro entonces, que de una correcta, completa y cuidadosa Ingeniería de Requisitos depende el éxito de un proyecto software, ya que ella evitará aquellos frecuentes desbordes en tiempo, costo y también la mala calidad de los productos software, generados por errores no detectados y corregidos a tiempo, rediseño, etc. Si bien, con un mínimo esfuerzo a corto plazo, esta etapa de la Ingeniería de Software podría "cumplirse", a largo plazo los cambios producidos por esta pobre especificación implicarán los desbordes mencionados y la ausencia de calidad del sistema.

Entonces, si es tan colosal el beneficio de la Ingeniería de Requisitos, porqué no desarrollar un modelo del proceso de la ingeniería de requisitos, para representar, analizar, conocer, predecir, evaluar, decidir, etc., sobre esta actividad y sus efectos en el proceso global del gestión de un proyecto de desarrollo de software. Porqué no sacar partido de esta etapa de la producción de sistemas (tomar a los requisitos como elementos de información y no como mera documentación de necesidades), para reducir costos, tiempos, esfuerzos, y aumentar conocimiento que de soporte a la toma de decisiones en el proceso de gestión.

I.3. *Objetivos*

I.3.1. *Generales*

- Proponer un marco conceptual-práctico enfocado en la gestión global de proyectos software que aborde el proceso de desarrollo desde la etapa de análisis hasta la etapa de prueba.
- Proporcionar un enfoque dinámico alternativo, complementario e integrador para la tarea de gestión de proyectos software que incluya los aspectos técnicos y socio-cultural-políticos de la organización y mejore la capacidad del proceso de desarrollo para generar productos de mayor calidad.
- Proveer un modelo de simulación de soporte a la toma de decisiones centrado en la gestión de requisitos como primer proceso clave para alcanzar un nivel de madurez de éxito repetible en los proyectos software.

I.3.2. *Específicos*

- Mejorar la eficiencia y eficacia de las actividades de gestión de proyectos software mediante la incorporación de modelos del proceso de Ingeniería de Requisitos a los modelos de soporte a la gestión.
- Proporcionar un modelo de proceso de Ingeniería de Requisitos, basado en la Dinámica de Sistemas, y su interfase para acoplarlo a un modelo de soporte a la gestión de proyectos software que integralmente proporcionen respuestas cuantitativas realistas referidas al proceso de desarrollo y al producto.
- Brindar una herramienta de simulación que permita ensayar diferentes alternativas de gestión de adaptar los recursos disponibles de dinero, tiempo y personas a lo largo de las etapas de análisis, diseño, codificación y prueba.

I.4. *Hipótesis*

El Modelo Dinámico Extendido de Gestión de Proyectos Software (MoDEGePS) ayuda a cuantificar con mayor precisión en las tareas de gestión de los proyectos de desarrollo de software.

I.4.1. *Operacionalización de las variables de la Hipótesis*

H: El MoDEGePS ayuda a cuantificar con mayor precisión en las tareas de gestión de los proyectos de desarrollo de software.

Si H ***entonces*** P1, P2 y P3

P1: El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de estimación.

P2: El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de planificación.

P3: El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de seguimiento y control.

VARIABLES	DIMENSIONES	INDICADORES
Tiempo	Planificación	Duración de las actividades y tareas individuales
		Tiempo necesario
Esfuerzo	Estimación	Esfuerzo necesario
		Costo necesario
Costo	Seguimiento y Control	Esfuerzo insumido
		Costo generado
Calidad		Productos generados (productividad y n° de errores)

Tabla I.1. Detalle de operacionalización de variables

I.4.2. *Comprobación de la Hipótesis*

Para la comprobación de la hipótesis se seleccionará el proyecto de software ya terminado (análisis post-mortem) DE-A. Los datos reales registrados para este proyecto se proporcionarán como entrada al modelo para simular el entorno de dicho proyecto, observar el comportamiento de la simulación y compararlo con el comportamiento real del proyecto.

Los resultados proporcionados por la simulación (estimados) serán comparados con los datos reales registrados para poder determinar si existe una subestimación o sobrestimación y el grado de desviación entre los valores comparados. Esto permitirá determinar qué tan exacto es el modelo en la reproducción de la historia del desarrollo real del proyecto de software seleccionado como caso de estudio.

El hecho de poder generar diferentes escenarios de simulación (diferentes valores para tamaño, esfuerzo, errores, etc.), permitirá analizar *"que hubiera pasado si..."*, de modo que se podrían evaluar diferentes políticas de gestión y sus efectos en el proceso de desarrollo de software, y a su vez se podrían inventariar qué reglas de gestión serían más o menos eficientes en determinadas condiciones de desarrollo.

El Proyecto DE-A

El proyecto DE-A (Dynamics Explorer-Attitude) fue un proyecto dirigido por el departamento de sistemas de la NASA, organización comprometida con el desarrollo de software para soportar el control y determinación de las acciones de naves espaciales.

Los requerimientos para el Proyecto DE-A fueron diseñar, implementar y probar un software que pueda procesar datos telemétricos y determinar las posiciones definitivas así como el soporte de control y determinación de posiciones en tiempo real para el satélite DE-A. Este satélite fue diseñado para estudiar la atmósfera, ionosfera y magnetosfera de la Tierra.

Este proyecto fue seleccionado como caso de estudio por satisfacer los dos siguientes criterios: 1). Mediana envergadura, es decir tamaño medio (entre 16-64 KSDI) y 2). Proyecto "típico", es decir desarrollado en un entorno familiar.

El tamaño del proyecto fue de 24.400 instrucciones fuente, fue terminado en 19 meses y consumió 2.222 hombre/días de esfuerzo.

CAPITULO II

.....

MARCOS REFERENCIALES

II.1. *Marco Conceptual*

Gestión: proceso que trata de optimizar el uso de recursos, principalmente los humanos, para conseguir unos objetivos con y a través de otras personas de la organización. Es un proceso de conducción del esfuerzo organizativo en la persecución de unos fines de la organización [21].

Proyecto: acción iniciada por una empresa en la que recursos humanos, financieros y materiales se organizan de una nueva forma para acometer un trabajo único, en el que, dadas unas especificaciones y dentro de unos límites de costos y tiempo, se intenta conseguir un cambio beneficioso definido por unos objetivos cualitativos y cuantitativos. Los elementos que componen esta definición implican que un proyecto conlleve una incertidumbre considerable y un riesgo, por lo tanto, su éxito dependerá en gran medida de una adecuada gestión [21].

Gestión de Proyecto Software: sistema de procedimientos, prácticas, tecnologías y conocimientos que facilitan las planificación, organización, gestión de recursos humanos, dirección y control necesarios para que el proyecto termine con éxito. Consiste en la utilización de técnicas y actividades de gestión requeridas para conseguir un producto software de alta calidad, de acuerdo con las necesidades de los usuarios, dentro de un presupuesto y con una planificación de tiempos establecidos previamente. Es el primer nivel del proceso de ingeniería del software. Es llamado nivel en lugar de paso o actividad, porque cubre todo el proceso técnico de desarrollo desde el principio al final, es decir se produce en paralelo con la totalidad del nivel técnico de construcción del software [21].

Gestión de Proceso Software: es el conjunto de técnicas y actividades que permiten una adecuada gestión de los procesos personales de los constructores y de los productos que participan en el proyecto [21].

Modelo: representación de un determinado aspecto de la realidad en un lenguaje específico. El modelo es un sistema artificial (en principio abstracto) que presenta el mismo comportamiento del sistema concreto original, o al menos una aceptable aproximación a él [3].

Modelos Dinámicos: constituyen una subclase de los modelos matemáticos. Son una representación de la conducta dinámica de un sistema. Aplican sus ecuaciones considerando cambios de tiempo, permitiendo de esta manera deducir los cambios de los atributos y actividades del sistema en función del tiempo. Son difíciles de resolver en forma analítica, por lo que es más frecuente resolverlos mediante métodos numéricos, como ser la simulación [10].

Un modelo dinámico está formalizado matemáticamente por un sistema de ecuaciones diferenciales que recogen las restricciones existentes entre magnitudes que evolucionan en el tiempo [31].

II.2. *Marco Teórico*

La siguiente organización y desarrollo del marco teórico se propone a partir de la búsqueda, clasificación, consulta y selección (realizada hasta la fecha) de la gran variedad de material bibliográfico existente (libros, tesis, revistas, resúmenes, etc.) que abordan desde distintos puntos de vista, la temática de la gestión de proyectos de desarrollo de software.

II.2.1. *El Proceso de Gestión de Proyectos de Desarrollo de Software [21]*

Para conseguir que un proyecto de software sea fructífero se debe comprender el ámbito del trabajo a realizar, los riesgos en los que se puede incurrir, los recursos requeridos, las tareas a llevar a cabo, los hitos que hay que recorrer, el esfuerzo (costo) a consumir y el plan a seguir. La gestión del proyecto de software proporciona este conocimiento. Empieza antes de que comience el trabajo técnico, continúa a medida que el proyecto software evoluciona desde el concepto hasta la realidad y culmina sólo, mucho tiempo después, en el momento en que se abandona el uso por parte del cliente del software producido.

Los ***elementos clave*** de la gestión de proyecto software son los siguientes:

a).- ***Comienzo del Proyecto***

Implica establecer el *ámbito*, los *objetivos*, considerar las *soluciones alternativas* e identificar las *restricciones* técnicas y de gestión. Sin esta información es imposible

obtener unas estimaciones de costo razonables, una identificación realista de las tareas del proyecto o un plan de trabajo adecuado que proporcione una indicación significativa del progreso.

El ámbito, también llamado alcance, identifica las *funciones* primordiales que debe llevar a cabo el software, el *rendimiento*, las *restricciones*, las *interfaces* y la *fiabilidad*.

Los objetivos identifican los fines globales del proyecto sin considerar cómo se llegará a esos fines.

Una vez entendidos los objetivos y el ámbito del proyecto, se han de considerar las soluciones alternativas que han de permitir a los gestores y a los desarrolladores seleccionar el mejor enfoque, dadas las restricciones impuestas por las fechas topes de entrega, los límites presupuestarios, la disponibilidad de personal, las interfaces técnicas y una multitud de otros factores.

b).- ***Métricas y Estimación***

En la mayoría de los trabajos técnicos, la medición y las métricas ayudan a entender el proceso técnico para desarrollar un producto, como el propio producto. El proceso se mide para mejorarlo. El producto se mide para intentar aumentar su calidad.

Las métricas son herramientas para poder estimar. Cuando se planifica un proyecto de software se tiene que obtener estimaciones del *esfuerzo* humano requerido (generalmente expresado en personas/mes), de la *duración* cronológica del proyecto (en fechas) y del *costo* (en valor monetario).

c).- ***Análisis de riesgos***

Consiste en una serie de pasos de control de los riesgos que permiten combatirlos: identificación de riesgos, cálculo de riesgos, priorización de riesgos, estrategias de control de riesgos, resolución de riesgos y supervisión de riesgos.

El análisis de riesgos consta de las cuatro actividades siguientes:

- *Identificación del riesgo:*

  Implica clasificar los riesgos dentro de las siguientes categorías:

  - *Riesgos de proyecto*: identifican potenciales problemas presupuestarios, de agenda, de personal, de recursos, de clientes y de requisitos, así como su impacto sobre el proyecto de software.
  - *Riesgos técnicos*: identifican potenciales problemas de diseño, implementación, verificación y mantenimiento.

- *Riesgos de negocios*: incluye a la construcción de un producto que nadie quiere, que no se ajusta a la estrategia global de producción de la empresa o que no se puede vender. La pérdida de presupuestos, de personal y de soporte de los gestores también se ubican en esta categoría.

- *Proyección del riesgo:*

  Intenta evaluar cada riesgo de dos formas: la probabilidad de que el riesgo sea real y las consecuencias de los problemas asociados con el riesgo suponiendo que aparezca.

  Las actividades de la proyección del riesgo son:

  - Establecimiento de una escala que refleje la probabilidad observada de un riesgo.
  - Definición de las consecuencias del riesgo.
  - Estimación del impacto del riesgo (determinado por su naturaleza, alcance y duración) en el proyecto y en el producto.
  - Anotación de la exactitud general de la proyección del riesgo.

- *Cálculo/Evaluación del riesgo:*

  Implica definir un nivel de referencia para el riesgo. El costo, la agenda y el rendimiento son tres niveles típicos de referencia. Es decir que hay un nivel de exceso de costo, de exceso de tiempo o de degradación del rendimiento o cualquier combinación de éstos que hará que se interrumpa el proyecto.

- *Gestión y supervisión del riesgo:*

  Puesto que gestionar un riesgo tiene costos adicionales para el proyecto, un paso importante en esta actividad es la evaluación de los beneficios conseguidos con los propios pasos de gestión del riesgo, comprobando que tengan un mayor peso que los costos asociados con su implementación. Es un típico análisis de costo-beneficio. En general los pasos de gestión están detallados en un Plan de Gestión y Supervisión de Riesgos.

  La supervisión del riesgo es una actividad de seguimiento con tres objetivos básicos: detectar la ocurrencia de un riesgo previsto, asegurar que los pasos de

gestión se estén aplicando correctamente y recopilar información para futuros análisis de riesgos.

d).- ***Planificación temporal del proyecto***

Esta actividad no difiere de la planificación de cualquier proyecto de ingeniería. Implica:

- *Identificar una serie de tareas del proyecto*
- *Establecer interdependencias entre tareas*
- *Estimar el esfuerzo asociado con cada tarea*
- *Asignar personal y otros recursos*
- *Crear una red de tareas*
- *Desarrollar un agenda de fechas*

e).- ***Sistema de control del proyecto***

Está basado en el principio de establecer un *bucle de retroalimentación* para asegurar que el proyecto está orientado a sus objetivos. Consiste en obtener información para tomar decisiones y asegurar a tiempo la detección y corrección de errores, controlando así la duración y presupuestos y minimizando los riesgos técnicos. Estos bucles actúan vía análisis de estado en informes de avances para comparar el progreso actual con los planes basados en las estimaciones.

Dentro de estos sistemas de control se encuentran:

- *Sistema de calidad:*

  Comprende dos grupos de actividades:

  - *Verificación, validación y pruebas*: Mediante la *verificación* se establece la correspondencia entre el producto software y sus especificaciones. Con la *validación* se identifica la robustez del producto para realizar su misión operativa. A través de las *pruebas* se comprueba el funcionamiento del código.
  - *Garantía de la calidad del software*: tiene como objetivo verificar la corrección de los procedimientos seguidos durante el desarrollo. Proporciona una visión independiente de los problemas de calidad, especialmente en la adopción de estándares y procedimientos al principio del proyecto

- *Sistema de gestión de la configuración:*

Es inevitable que la definición de un producto esté sujeta a la presión de cambios continuos durante su ciclo de vida, a la corrección de errores, a la introducción de mejoras y requisitos en evolución debido a cambios del mercado. La gestión de la configuración proporciona la disciplina requerida para prevenir el caos del cambio incontrolado.

II.2.2. *Enfoque Tradicional de Gestión de Proyectos de Desarrollo de Software*

a).- ***Medidas, Métricas y Estimación [26]***

Dentro del contexto de la Ingeniería del Software, una *medida* proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto. La *medición* es el acto de determinar una medida.

Según IEEE Standard Glosary of Software Engineering Terms, una *métrica* es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Un *indicador* es una métrica o combinación de métricas que proporciona una visión profunda del proceso de software, del proyecto de software o del producto en sí.

La primera aplicación de métricas de proyectos en la mayoría de los proyectos de software ocurre durante la estimación. Las métricas recopiladas de proyectos anteriores se utilizan como base desde la que se realizan las estimaciones del esfuerzo y del tiempo para el actual trabajo de software. A medida que avanza un proyecto, las medidas del esfuerzo y del tiempo consumido se comparan con las estimaciones originales (y la planificación del proyecto). El gestor de proyectos utiliza estos datos para supervisar y controlar el avance.

- ***Clasificación de las Métricas del Software [21]***

Existen diferentes clasificaciones de métricas del software. A los efectos del presente trabajo de investigación se selecciona la siguiente:

- ***Métricas de Productos:*** medidas del producto software durante cualquier fase de su desarrollo, desde los requisitos hasta la instalación. Entre otras, se

incluyen en estas medidas la complejidad del diseño, el tamaño del producto final o el número de páginas de documentación producida.

Dentro de estas métricas, existe una sub-clasificación en las siguientes categorías, que son de particular interés para realizar el proceso de estimación del software:

Métricas del Tamaño:

Dentro de este tipo de métricas, se encuentran:

Líneas de Código (LOC – Lines of Code): es la medida de longitud del código fuente más utilizada. Puede calcularse de distintas maneras, dependiendo si se incluyen en la medida las líneas en blanco, los comentarios, las secuencias no ejecutables, las instrucciones múltiples por línea y las múltiples líneas por instrucción. Para que esta medida refleje la longitud real del programa se toma: LOC = NCLOC + CLOC, donde NCLOC incluye todas las líneas de código que no tienen comentarios (cabeceras, declaraciones e instrucciones ejecutables y no ejecutables), y CLOC es el número de líneas de comentarios.

Para utilizar la medida LOC en la estimación de esfuerzo o productividad, atendiendo a los conceptos de reutilización, costos fijos y tareas que se realizan que no producen instrucciones, LOC se puede medir en términos de *número de bytes de almacenamiento requerido* o del *número de caracteres en el texto del programa.*

Puntos de Función (FP – Function Point): miden el software cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. La aplicación de esta técnica comprende los siguientes pasos:

- *Definición de los límites del sistema:*

  El límite se utiliza para definir el alcance del sistema y ayuda a identificar los parámetros externos. Existen tres versiones de los límites del sistema, dependiendo de la utilización que quiera realizarse de la técnica. Estas versiones son: *Límite del producto* (cuando la cuenta de PF se realiza al final del

desarrollo del proyecto), *Límite inicial del proyecto a desarrollar* (el conteo de los PF se deriva de los requisitos de un sistema que aún no existe) y *Límite del proyecto de mejora* (el conteo se realiza sobre un sistema que ya existe pero al que se le han realizado adiciones, modificaciones o anulaciones de funcionalidades).

- *Definición de parámetros:*

  Es la determinación de los componentes del sistema que pueden ser clasificados como tipos de funciones y son de dos clases: datos o transacciones.

  Dentro de los *tipos de funciones de datos* se incluyen los siguientes parámetros:

  - *Ficheros lógicos internos*: grupo de datos lógicamente relacionados, identificables por los usuarios o información de control, mantenidos y utilizados dentro de los límites de la aplicación.
  - *Ficheros interfaces externos*: grupos de datos relacionados lógicamente, identificables por los usuarios o información de control utilizada por la aplicación pero mantenida por otra aplicación.

  Dentro de los *tipos de funciones transacción* se incluyen los siguientes parámetros:

  - *Entradas externas*: datos o información de control que se introducen en la aplicación desde afuera de sus límites. Estos datos mantienen un fichero lógico interno.
  - *Salidas externas*: datos o información de control que sale de los límites de la aplicación. Esta salida debe ser considerada única si tiene un formato único o si el diseño lógico requiere un proceso lógico distinto de otras salidas del mismo formato.

- *Consultas externas*: requisitos de información a la aplicación en una combinación única de entrada/salida que se obtiene de una búsqueda de datos, no actualiza un fichero lógico interno y no contiene datos derivados.

- *Valoración de la complejidad:*

  Para cada uno de los parámetros se identifica su complejidad como baja, media o alta. Para las entradas, salidas y consultas, se puede evaluar su complejidad en función del número de campos que contengan y del número de ficheros a los que hagan referencia. Para los ficheros, por el contrario, su complejidad vendrá dada en función del número de registros y campos que tengan.

  Una vez definida la complejidad de cada parámetro se multiplica por un peso ya definido. La suma total de estos valores da los puntos de función (sin ajustar).

- *Análisis de las características generales del sistema:*

  Se realiza para ajustar el total de puntos de función obtenidos. Las características que se analizan son: comunicación de datos, funciones distribuidas, rendimiento, configuraciones fuertemente utilizadas, frecuencia de transacciones, entrada on-line de datos, diseño para la eficiencia del usuario final, actualización on-line, procesos complejos, utilización en otros sistemas, facilidad de instalación, facilidad de operación, instalación de múltiples sitios, facilidad de cambio.

  A cada una de estas características se da una valoración de 0 a 5 según una guía proporcionada en la técnica y luego la suma de esos valores constituye el *grado de influencia* (TDI Total Degree of Influence). El TDI se utiliza para calcular el *factor de ajuste* (AF - Adjustment Factor) mediante la siguiente fórmula:

$$AF = (TDI * 0,01) + 0,65$$

El valor final de los puntos de función ajustados será:

$$FPA = PF * AF$$

Bang: se calculan utilizando las funciones primitivas obtenidas a partir de las especificaciones formales del software. El modelo proporciona diferentes fórmulas y criterios para distinguir entre sistemas que utilizan algoritmos complejos frente a aquellos sistemas orientados a datos. No se proporciona mayor detalle de esta métrica por no estar muy extendida.

Métricas de Calidad: [26]

Hay muchas medidas de la calidad del software pero los indicadores más útiles para el equipo de proyecto son las siguientes:

Corrección: grado en que el software lleva a cabo su función requerida. La medida más común de corrección son los *defectos*, en donde *defecto* se define como una falta verificada de conformidad con los requisitos.

Facilidad de mantenimiento: facilidad con que un programa se puede *corregir* si se encuentra un error, *adaptar* si su entorno cambia o *mejorar* si el cliente desea un cambio de requisitos. Como no se puede medir directamente se deben utilizar medidas indirectas como el *tiempo medio de cambio (TMC)*, es decir, el tiempo que se tarda en analizar la petición de cambio, en diseñar una modificación adecuada, en implementar el cambio, en probarlo y en distribuir el cambio a todos los usuarios.

Integridad: este atributo mide la habilidad del sistema para resistir ataques, tanto accidentales como intencionados, contra su seguridad. Para medir este atributo, se utilizan otros dos atributos adicionales: *amenaza* que es la probabilidad de que un "ataque" de un tipo determinado ocurra en un tiempo determinado y *seguridad* que es la probabilidad de que se pueda repeler el ataque de un tipo determinado.

Facilidad de uso: cuantifica la amigabilidad con el usuario y se puede medir en función de cuatro características: *habilidad intelectual y/o física requerida para aprender el sistema, tiempo requerido para llegar a ser moderadamente eficiente en el uso del sistema, aumento neto en productividad en la utilización moderada y efectiva del sistema, valoración subjetiva de la disposición de los usuarios hacia el sistema.*

- ***Métricas de Proceso:*** medidas del proceso de desarrollo del software tales como tiempo de desarrollo total, esfuerzo en hombres/días o hombres/meses de desarrollo de software, el número y tipo de recursos empleados (personas, maquinas, etc.), el costo del proceso, tipo de metodología utilizada o nivel medio de experiencia de los programadores.

b).- ***Técnicas de Estimación [21]***

Para la estimación existen cuatro técnicas básicas y comunes:

- ***La opinión de los expertos****:* se basa en la experiencia profesional de los que participan en el proyecto de estimación. Es una técnica informal que no ha otorgado buenos resultados.

- ***La analogía****:* es una aproximación más formal que la experiencia de los expertos y se basa en la comparación directa de uno o más proyectos pasados. La estimación inicial se ajusta dependiendo de las diferencias entre el proyecto

pasado y el nuevo. Para poder utilizarla es necesario disponer de una base de datos histórica de proyecto finalizados para poder realizar la comparación.

- ***La descomposición**:* consiste en la descomposición de un producto en componentes más pequeños, o descomponer un proyecto en tareas de nivel inferior. La estimación se hace a partir del esfuerzo requerido para producir los componentes más pequeños o para realizar las tareas de nivel inferior. La estimación global del proyecto resultará de sumar las estimaciones de los componentes. Al igual que en la técnica anterior se requiere de una base de datos histórica. Generalmente no están disponibles.

- ***Las ecuaciones de estimación**:* son fórmulas matemáticas que establecen la relación de algunas medidas de entrada (que normalmente es la medida del tamaño del producto) y determinan el esfuerzo que se requerirá.

Dadas las desventajas de las tres primeras técnicas, cuando se comienza a realizar el proceso de estimación se utiliza algún *método o modelo*, es decir, se emplea la cuarta técnica. Dentro de este grupo se encuentran los siguientes:

- Modelos de Estimación [21]

<u>*Modelos Estadísticos*</u>*:* Desarrollado por C. E. Walston y P. C. Félix. A partir de sesenta proyectos terminados establecieron el siguiente modelo simple de cálculo del *esfuerzo* de desarrollo de software. $E = 5{,}2\ L^{0{,}91}$ donde L es el número de líneas de código en miles.

<u>*Modelos basados en teorías*</u>*:* Desarrollado por Putnam. Este modelo asume una distribución específica del esfuerzo a lo largo de la vida de un proyecto de desarrollo de software. La relación entre el tamaño del producto software y el tiempo de desarrollo es la siguiente: $K = L^3 / C^3 T^4$ donde L es el número de instrucciones fuente producidas, K el esfuerzo durante todo el ciclo de vida en personas/años, T el tiempo de desarrollo en años y C una constante

dependiente de la tecnología (C=2000 para un entorno de desarrollo pobre; C=8000 para un entorno bueno de desarrollo y C=11000 para un entorno de desarrollo excelente).

Modelos compuestos: Utilizan una combinación de intuición, análisis estadístico y juicio de expertos.

- ***COCOMO (Böehm):***

Es el modelo más conocido y sólidamente documentado para la estimación de costos. Existen tres modos de desarrollo: *orgánico, semilibre y rígido* que ayudan a determinar la dificultad del proyecto según las características de la aplicación y del entorno de desarrollo. A cada uno de estos modelos se le puede aplicar tres niveles de estimación de modelo: *básico, intermedio y detallado*.

Lo que distingue a cada *modo* es el mayor o menor grado/nivel en que se dan las siguientes características: *tamaño del equipo de desarrollo, experiencia de los miembros del equipo de desarrollo, presión o exigencia por parte de los usuarios en el cumplimiento de las especificaciones, el tamaño del producto, etc.*

La ecuación de *esfuerzo* de desarrollo es: $MM = a * S^b * m$ y la de *duración* es: $TDEV = a * MM^b$, donde S es el número de líneas de código, a y b son constantes determinadas para cada modo y nivel del modelo y m es un factor de ajuste determinado que resulta de la productoria del valor asignado (en un rango de muy bajo a extra alto) a cada uno de 15 atributos de 15 factores que inciden en el costo (para el nivel intermedio) y son: *fiabilidad requerida del software, tamaño de la base de datos, complejidad del producto, limitaciones en el tiempo de ejecución, limitaciones de memoria principal, volatilidad de la máquina virtual, frecuencia de cambio en el modelo de explotación del ordenador, capacitación de los analistas, experiencia en aplicaciones, capacitación de los programadores, experiencia en la máquina virtual, experiencia en el lenguaje de programación, prácticas modernas de programación, uso de*

herramientas para el desarrollo de software, limitaciones en la planificación.

La Tabla II.1 resume los valores para las constantes *a* y *b* de las ecuaciones para el esfuerzo medio en meses/hombres (MM) y duración en meses (TDEV) para cada modo y nivel del modelo.

		Modelo Básico		***Modelo Intermedio***	
Orgánico	***MM***	a=2,4	b=1,05	a=3,2	b=1,05
	TDEV	a=2,5	b=0,38	a=2,5	b=0,38
Semilibre	***MM***	a=3,0	b=1,12	a=3,0	b=1,12
	TDEV	a=2,5	b=0,35	a=2,5	b=0,35
Rígido	***MM***	a=3,6	b=1,20	a=2,8	b=1,20
	TDEV	a=2,5	b=0,32	a=2,5	b=0,32

Tabla II.1. Valores de las constantes de las ecuaciones de Esfuerzo y Duración según modo y nivel del modelo

- ***SOFTCOST (Tausworthe):***

Estima el costo del software utilizando 68 parámetros cuyos valores se deducen de respuestas del usuario a 47 preguntas a cerca del proyecto.

- ***SPQR – Software Productivity, Quality and Reliability (Capers Jones):***

Está basado en 45 factores que influyen en el costo y productividad del desarrollo de software. Requiere responder a más de 100 preguntas relacionadas con el proyecto para formular los parámetros de entrada necesarios en el cálculo de los costos y los planes. Este modelo no está bien documentado.

- ***CPMO (Thebaut):***

Proporciona una ecuación general para el cálculo del esfuerzo requerido en grandes proyectos. $E = a + bS + cP^d$ donde *a, b, c* y *d* son

constantes determinadas mediante regresión, a partir de datos empíricos, *S* es el tamaño del programa en miles de LOC y *P* es el nivel medio de personal durante el ciclo de vida del proyecto.

- ***ESTIMATICS (Rubin):***

Proporciona la estimación del esfuerzo total, requisitos de personal, costo, riesgo y efecto sobre la cartera de proyectos. El inconveniente de este modelo es que el autor no ha hecho públicas las ecuaciones para realizar los cálculos de las estimaciones.

Si bien a lo largo de esta sección del marco teórico se presentaron y desarrollaron las diversas clases de *métricas, técnicas de estimación y modelos empíricos de estimación*, es importante aclarar que a los efectos de esta investigación se utilizarán, como entrada al modelo a desarrollar, ciertos valores estimados obtenidos a partir de las técnicas y modelos que se han impuesto más sólidamente en la disciplina de la Ingeniería de Software. Tal es el caso de la técnica de Puntos de Función y COCOMO.

II.2.3. *Enfoque Alternativo de Gestión de Proyectos de Desarrollo de Software*

a).- ***Dinámica de Sistemas (DS) [20]***

En 1961, Jay W. Forrester publicó su obra *Industrial Dynamics* que marca el comienzo de la "técnica DS" como procedimiento de estudio y simulación del comportamiento de sistemas sociales. En 1969, se publica la obra *Dinámica Urbana*, en la que se muestra cómo el "modelado DS" es aplicable a sistemas de ciudades.

En el momento actual, la DS o Simulación Dinámica, es una técnica de uso generalizado para modelar y estudiar el comportamiento de cualquier clase de sistemas con tal que éste tenga las características de existencia de retardos y bucles de realimentación.

Los modelos DS constituyen un grupo particular de los modelos matemáticos, por lo tanto, gozan de todas las características generales de éstos. Sin embargo tienen unas peculiaridades que los identifican. En correspondencia con las características estructurales y funcionales de los modelos matemáticos, se pueden establecer otras para los modelos DS:

1º). *Variables de estado o nivel:* la denominación de nivel se toma del símil hidrodinámico. Están referidas a un instante de tiempo.

2º). *Variables de flujo:* son las variables que afectan el comportamiento de las variables de estado (haciendo que crezcan o disminuyan). Éstas variables están referidas a un período de tiempo.

3º). *Variables auxiliares:* son magnitudes que ayudan a explicar los valores de los flujos. Pueden estar referidas a un instante de tiempo o a un período de tiempo.

4º). *Variables exógenas o independientes:* son variables externas al sistema, que actúan sobre el comportamiento de éste. Desde el punto de vista de la dimensión temporal estas variables pueden estar referidas a un instante de tiempo o a un período de tiempo.

Los niveles, los flujos y las variables auxiliares son *variables endógenas o dependientes.*

5º). *Parámetros:* son las magnitudes constantes del sistema, es decir permanecen invariables a lo largo del período de estudio del sistema.

6º). *Canal de material:* representa la acción de un flujo sobre un nivel.

7º). *Canal de información:* representa las interrelaciones entre las variables y entre variables y parámetros.

8º). *Fuentes y sumideros:* indican la procedencia o destino de materiales que no son de interés para el estudio del modelo.

9º). *Relación no lineal:* para indicar que entre dos variables existe una relación no lineal.

10º). *Retardos:* para señalar que la transmisión de información no es inmediata, sino que se lleva a cabo en período de tiempo mayor al elegido como unidad temporal de análisis.

11º). *Variables predeterminadas:* están dadas en períodos o instantes de tiempo anteriores al horizonte cero.

Etapas para elaborar un modelo DS

1). *Descripción del sistema, identificación de elementos y relaciones fundamentales.*
2). *Construcción del Diagrama Causal.*
3). *Definición precisa de cada magnitud: código de variables.*
4). *Construcción del Diagrama de Forrester.*
5). *Construcción del Sistema de Ecuaciones.*
6). *Calibrado.*
7). *Análisis de sensibilidad.*
8). *Evaluación del modelo: contrastado.*
9). *Utilización del modelo: escenarios e imágenes simuladas.*

A posteriori, en la Sección "Marco Metodológico" se detallará cada etapa.

- ***Modelos de DS para la Gestión de Proyectos***

Es importante destacar la utilidad de los modelos de la DS para entender las interacciones entre las variables relevantes en la gestión de un proyecto y formular políticas de decisión efectivas para lograr el éxito. Los errores cometidos durante la ejecución generan repetición del trabajo, actividades no previstas, ajustes al cronograma, deterioro de la calidad e incluso revisión del alcance del proyecto. Para entender y anticipar estas situaciones no deseadas, los modelos de DS permiten visualizar un amplio rango de situaciones antes de tomar las decisiones pertinentes. Además, permite visualizar múltiples ciclos de retroalimentación y manejar relaciones no lineales. Por otra parte, combina el manejo de variables "duras" y "blandas", lo cual no es posible con las herramientas tradicionales de la Investigación de Operaciones.

Además la DS permite que los proyectos puedan ser simulados sin el riesgo de utilizar recursos de manera inadecuada y además con el beneficio de adquirir el aprendizaje necesario para aplicarlo posteriormente en la fase de inversión o ejecución [15].

Las aplicaciones de la DS en la Ingeniería del Software son múltiples. Entre ellas se destaca principalmente su aplicación a la investigación de nuevas políticas de desarrollo, que se ve facilitada por la capacidad de simulación. También se

destacan sus aplicaciones dirigidas a la formulación de una metodología formal, que permita estandarizar el proceso de desarrollo de software, los análisis de proyectos ya terminados (análisis post-mortem) y la monitorización y seguimiento continuo de los proyectos en desarrollo.

Por tanto, el marco de la DS ofrece las bases para construir una teoría común para los procesos de desarrollo de software. La elaboración de modelos dinámicos puede constituir una metodología formal, según la cual se pueden expresar los conocimientos sobre el sistema. Además, el proceso de construcción del modelo, por sí mismo, obliga a los investigadores a tener un alto conocimiento de cuáles son los parámetros claves que influyen en el comportamiento del sistema y cómo se relacionan entre sí constituyendo lazos de realimentación.

Por otro lado, el potencial de los modelos de simulación para la formación y el entrenamiento de los directores de proyectos es manifiesto: los entornos de simulación posicionan a los directores frente a situaciones reales que pueden encontrar en la práctica y les permite adquirir experiencia sin correr riesgos. La disponibilidad de un modelo dinámico, que simule el comportamiento o algún aspecto concreto de una organización, y un entorno de simulación potente, constituyen una herramienta fundamental en la toma de decisiones de dicha organización.

La aplicación de la DS a la gestión de los proyectos de desarrollo de software nos permite considerarlos como sistemas dinámicos socio-tecnológicos complejos, cuya evolución temporal vendrá dada tanto por su estructura interna como por las políticas de dirección empleadas, y las relaciones establecidas entre el personal técnico que participa en el proyecto. La anterior afirmación permite desarrollar modelos multivariables dinámicos para describir los procesos mentales seguidos por los gestores de proyectos en la toma de decisiones, en muchas ocasiones basados exclusivamente en la experiencia y, lo que es más importante, poder simular el comportamiento del modelo en base a las decisiones tomadas [28].

- El Modelo de Abdel-Hamid y Madnick [1]

Dado que la mayor deficiencia en muchas de las investigaciones sobre gestión de proyectos de software es la falta de capacidad para realizar inferencias acerca del comportamiento del sistema como una totalidad

socio-técnica desde el conocimiento de los micro-componentes tales como la planificación y productividad, éste modelo se construye bajo el entendimiento de esos micro-componentes.

El modelo tiene dos características claves que lo distinguen de muchos otros modelos en el área de la Ingeniería del Software:

a).- Es *integrador*: en el sentido que integra las múltiples funciones del proceso de desarrollo de software, incluyendo tanto las funciones de *gestión* (planificación, control y asignación de personal) como las de *producción* de software que constituyen el ciclo de vida de desarrollo de software (diseño, codificación, revisión y prueba).

b).- Es un modelo de *Dinámica de Sistemas*: fundado en las distintas premisas de la filosofía de la DS, se establecieron dos bases principales para operacionalizar la técnica y sobre las que se apoya este modelo: una es el uso de los sistemas de feedback de información (para modelar y entender la estructura del sistema) y la otra es el uso de la simulación por computadora (lo que permite experimentar superando las barreras de la intuición y tiempo para estudiar y predecir implicaciones dinámicas).

Bajo estas características, se está ante un modelo que usa los principios de feedback de la DS y la simulación para estructurar y clarificar la compleja red de variables que interactúan dinámicamente y que caracterizan la conducta de un sistema socio-técnico ya que integra las funciones tanto de nivel de gestión como del nivel de producción.

El límite del modelo

El foco de interés de este modelo está puesto en las fases de desarrollo de la producción de software, extendiéndose únicamente desde la fase de diseño hasta la última fase de desarrollo llamada fase de prueba. Queda claro entonces que excluye las fases de análisis y mantenimiento. El autor justifica esto aludiendo que su interés está puesto en las acciones, políticas y decisiones de los miembros del grupo de desarrollo y no de los usuarios. Este límite se amplía con el desarrollo del presente trabajo de investigación, al incluir en el modelo la fase de análisis estrechamente relacionada con los usuarios, ya que es claro que las acciones, políticas y decisiones de los

usuarios afecta las acciones, políticas y decisiones de los miembros del equipo de desarrollo de software.

La estructura del modelo

El modelo consiste de cuatro subsistemas mayores, *gestión de recursos humanos, planificación, control y producción de software,* junto con los diferentes flujos que los conectan. Sintéticamente esta estructura se muestra en la Figura II.1.

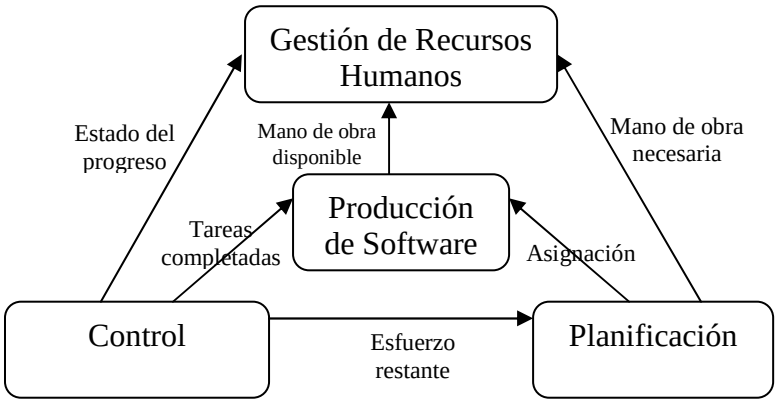


Figura II.1. Estructura del Modelo de Abdel-Hamid y Madnick para la Gestión de Proyectos Software

Subsistemas del Modelo:

Subsistema de Gestión de Recursos Humanos: Comprende la contratación, entrenamiento, asignación y rotación de los recursos humanos.

Subsistema de Producción de Software: Considera las siguientes cuatro actividades de producción como primarias: desarrollo, garantía de calidad, re-trabajo y prueba. El desarrollo comprende tanto el diseño como la codificación. Cuando el software es desarrollado, es también revisado para detectar cualquier error de diseño o código. Los errores detectados a través de la garantía de calidad son re-trabajados.

No todos los errores serán detectados y re-trabajados, algunos serán pasados por alto hasta ser descubiertos en la fase de prueba.

Subsistema de Control: Determina el estado del proyecto. Implica realizar una comparación del estado actual del proyecto y el estado en que debería encontrarse de acuerdo a lo planificado. Se mide a través del número de tareas completadas.

Subsistema de Planificación: Comprende la realización de estimaciones iniciales para comenzar el proyecto y que luego se revisan (cuando es necesario) a lo largo de la vida del proyecto.

II.2.4. *Análisis de Sistemas [26]*

En el marco de la Ingeniería del producto (considera al software como solución de un problema planteado), la meta es traducir el deseo de un cliente, un conjunto de capacidades definidas, en un producto operativo. Para conseguir esta meta se debe crear una arquitectura y una infraestructura. La *arquitectura* comprende cuatro componentes de sistemas distintos: software, hardware, datos (base de datos) y personas. La *infraestructura* incluye la tecnología requerida para unir los componentes y la información que se emplea para dar soporte a los componentes.

Una visión general de tal arquitectura e infraestructura se consigue a través del *análisis del sistema.*

En esta etapa de ingeniería, los requisitos generales del producto se obtienen del cliente. Estos requisitos comprenden necesidades de información y control, funcionalidad, comportamiento y rendimiento general del producto, diseño, restricciones de la interfaz y otras necesidades especiales. Una vez que se conocen estos requisitos, la misión del análisis de sistemas es asignar funcionalidad y comportamiento a cada uno de los cuatro componentes de la arquitectura mencionados anteriormente.

Una vez que se ha hecho la asignación, comienza la *ingeniería de componentes* que es, de hecho, un conjunto de actividades concurrentes que se dirigen separadamente a cada uno de los componentes del sistema: la ingeniería del software, la ingeniería del hardware, la ingeniería humana e ingeniería de bases de datos. Cada una de estas disciplinas de ingeniería toma una vista de dominio específica, pero también deben establecer y mantener

una comunicación activa entre ellas. También es parte del análisis de sistemas establecer los mecanismos de interfaz que permitan tal comunicación.

Luego sigue una *visión de elemento*, que es la disciplina de ingeniería aplicada a la asignación de componentes. Para la ingeniería del software, esto significa *actividades de modelado del análisis y diseño* y *actividades de construcción e integración* que comprenden generación de código, pruebas y actividades de soporte. El modelado de análisis asigna requisitos a las representaciones de datos, funciones y comportamiento.

La actividad de análisis de sistemas se corresponde con la primera de tres fases genéricas que [26] asocia al trabajo de ingeniería de software y que la denomina *fase de definición*.

En esta fase, el desarrollo del software se centra en el *qué*, es decir, se identifica *qué* información ha de ser procesada, *qué* función y rendimiento se desea, *qué* comportamiento del sistema y *qué* interfaces van a ser establecidas, *qué* restricciones de diseño existen y *qué* criterios de validación se necesitan para definir un sistema correcto. Es fundamental en esta fase identificar los requisitos clave del sistema y del software. Como lo expresa [26], para el éxito de un desarrollo de software es esencial una comprensión total de los requisitos del software. No importa lo bien diseñado o codificado que esté el programa si no se ha analizado correctamente, pues defraudará al usuario y frustrará a quien lo desarrolla.

Ahora bien, ¿qué son los requisitos?

Según [35], se pueden definir los requisitos desde el punto de vista del usuario o de quien desarrolla el producto.

Los *requisitos del usuario* son las condiciones o capacidades necesarias para que un usuario pueda resolver un problema o alcanzar un objetivo.

Los *requisitos para equipo de desarrollo* son las condiciones o capacidades que debe reunir un sistema para satisfacer un contrato estándar o cualquier otro documento impuesto formalmente.

Navarro, Antonio [24] define una clasificación de requisitos como sigue:

- *Requisitos funcionales:* describen los servicios que proporciona el sistema, la respuesta del sistema ante determinadas entradas y el comportamiento del sistema en situaciones particulares.

- *Requisitos no funcionales*: son restricciones de los servicios o funciones que ofrece el sistema (tiempo, proceso de desarrollo, etc.). Dentro de esta clase hay tres tipos:
  - *Requisitos del productos*: especifican el comportamiento del producto (prestaciones memoria, tasa de fallos, etc.)
  - *Requisitos organizativos*: se derivan de las políticas y procedimientos de las organizaciones de los clientes y desarrolladores (estándares de proceso, lenguajes de programación, etc.)
  - *Requisitos externos*: se derivan de factores externos al sistema y al proceso de desarrollo (requisitos legislativos, éticos, etc.)
- *Requisitos de dominio*: se derivan del dominio de la aplicación y reflejan características de dicho dominio. Pueden ser funcionales o no funcionales.

a).- ***Ingeniería de Requisitos***

Según [22], la Ingeniería de Requisitos es la disciplina de la Ingeniería de Software donde se identifica el propósito, la dirección y el alcance del sistema. Consiste en un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requerimientos del sistema siguiendo un determinado estándar. Los requisitos constituyen el enlace entre las necesidades reales de los clientes, usuarios y otros participantes vinculados al sistema (stakeholders).

El proceso de Ingeniería de Requisitos, es un conjunto de actividades que se siguen con el objetivo de descubrir, modelar, validar y mantener un documento de requisitos. Este proceso debe "lidiar" con diferentes puntos de vista, usar una combinación de técnicas, herramientas y personas.

Para [24], el proceso de Ingeniería de Requisitos consta de las siguientes fases, cada una de las cuales produce resultados diferentes:

- *Estudio de Viabilidad:*

  Implica realizar una estimación sobre si es posible resolver el problema del usuario con las tecnologías software y hardware disponibles. Este estudio permite decidir si el sistema es rentable y puede desarrollarse cumpliendo las restricciones económicas. El producto de esta fase es el informe de viabilidad.

- *Análisis de Requisitos:*

  Se derivan los requisitos de la aplicación mediante la observación de sistemas existentes, discusiones con usuarios potenciales, analistas y demás implicados. Se pueden desarrollar modelos y/o prototipos del sistema. Se genera un documento introductorio con los requisitos del usuario.

  A su vez el proceso de análisis de requisitos incluye:

  - *Comprensión del dominio*: los analistas deben comprender el dominio de la aplicación.
  - *Recolección de requisitos:* interacción con stakeholders del sistema para descubrir sus requisitos.
  - *Clasificación:* recolectar el conjunto no estructurado de requisitos y organizarlos en agrupaciones coherentes.
  - *Resolución de conflictos:* resolver los conflictos que pudieran producirse entre los requisitos de los stakeholders
  - *Priorización:* interacción con los stakeholders para identificar los requisitos más importantes del sistema.
  - *Comprobación de requisitos:* comprobar los requisitos para determinar si son completos y consistentes con los deseos de los stakeholders.

- *Especificación de Requisitos:*

  Se fija una descripción detallada y precisa de los requisitos del sistema, de tal manera que sirva de base para un contrato entre el cliente y el desarrollador del software. El objetivo es obtener una especificación de requisitos del sistema. Pueden utilizarse distintos lenguajes para especificar los requisitos: lenguaje natural, lenguaje natural estructurado, lenguaje de descripción de diseño, lenguaje de especificación de requisitos, notaciones gráficas, especificaciones matemáticas. El producto derivado de esta fase son los requisitos del sistema.

- *Validación de Requisitos:*

  Se encarga de comprobar si los requisitos se ajustan a los deseos de los stakeholders. Si la validación es inadecuada, se propagarán errores al diseño e implementación. Evidentemente, esto tiene una fuerte repercusión sobre el costo. El objetivo es obtener la especificación de requisitos software (ERS)

Hay distintos tipos de comprobaciones en el proceso de validación de requisitos:

- *Validez:* Comprobar la necesidad de las funciones definidas en la ERS.
- *Consistencia:* Comprobar que no hay restricciones contradictorias o distintas descripciones de la misma función.
- C*ompletitud:* Comprobar que todos los requisitos y restricciones están incluidos en la ERS.
- *Realismo:* Comprobar que se pueden implementar los requisitos con las tecnologías disponibles.
- *Verificabilidad:* Comprobar que los requisitos pueden ser verificados con el fin de evitar problemas entre clientes y desarrolladores.

II.3. *Marco Metodológico*

A partir del análisis de la bibliografía consultada sobre el área de la gestión de proyectos de desarrollo de software, se pudo determinar que aunque ha sido reconocida la necesidad de la gestión proyectos y a partir de ello se han desarrollados algunas herramientas de apoyo a esta actividad, se siguen presentando casos en los que el esfuerzo, los costos y los tiempos exceden a los previstos.

En los métodos y modelos de gestión desarrollados y utilizados actualmente, se ha encontrado una doble falencia, por un lado, su base estática y empírica (en los tradicionales) y por otro, su aplicación parcializada -solo considera ciertas etapas- al proceso de desarrollo de software (en los dinámicos).

Atento a ello, es que se propone un nuevo modelo de carácter dinámico, que tiene base en la Dinámica de Sistemas para la gestión de proyectos de desarrollo de software, el cual, por un lado, contemple en el proceso de desarrollo las etapas de análisis, diseño, codificación y prueba, las relaciones entre ellas y las repercusiones de estas relaciones en el proyecto global, y por otro, mediante herramientas de simulación, sea vehículo de experimentación que proporcione resultados que se puedan utilizar como guía para la toma de decisiones que conduzcan al éxito del proyecto.

Por lo tanto el desarrollo del modelo propuesto se hará bajo los lineamientos de la *Metodología Dinámica de Sistemas.* Las características generales de esta metodología ya

fueron comentadas en el marco teórico, resta entonces describir en forma genérica las etapas a través de las cuales se construirá el modelo.

Etapas para elaborar un modelo DS [20]

1). *Descripción del sistema, identificación de elementos y relaciones fundamentales*

En esta etapa se trata de precisar el esqueleto básico del modelo. De acuerdo con los fines y los límites, se señalan y seleccionan aquellos elementos fundamentales y también se establecen las relaciones entre ellos, al menos las más evidentes. En un proceso de refinamiento se desagregarán o rechazaran elementos y relaciones.

2). *Construcción del Diagrama Causal*

Es una formalización de la etapa anterior. A cada elemento se le da un nombre y se representa con un punto. Las relaciones entre los elementos se indican con flechas y deben llevar un signo. El signo será positivo (+) si en esa relación las variables se mueven temporalmente en el mismo sentido. Si se mueven temporalmente en sentido opuesto, el signo será negativo (-). Este diagrama permite identificar los bucles de retroalimentación que también pueden ser positivos y negativos. Si todas las relaciones del bucle son positivas, el bucle también será positivo. Si todas las relaciones son negativas y el número de relaciones es impar el bucle es negativo, y si es par, el bucle será positivo. Esta última regla también es válida cuando en el bucle hay relaciones positivas y negativas.

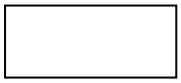
3). *Definición precisa de cada magnitud: código de variables*

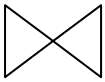
Implica construir una tabla de variables que contiene para cada una de ellas la siguiente información:

* Primera columna: número de orden
* Segunda columna: símbolo nemotécnico
* Tercera columna: descripción
* Cuarta columna: tipo
* Quinta columna: magnitud (unidad de medida) con la cual se mide

4). *Construcción del Diagrama de Forrester*

No es otra cosa que un diagrama causal una vez que se han clasificado las magnitudes de acuerdo con la convención de la DS que se muestra a continuación:

Variable de Nivel: 

Variable de Flujo: 

Variable Auxiliar: 

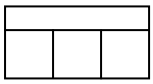
Variables Exógenas: 

Parámetros: 

Canal de Material:

Canal de Información:

Fuente o Sumidero: 

Retardo: 

5). *Construcción del Sistema de Ecuaciones*

Todas las *ecuaciones de nivel* son definicionales y tiene la forma general:

$$N(K) = N(J) + DT * [FE(JK) - FS(JK)]$$ donde:

N(K) es el valor del nivel en el instante T

N(J) es valor del nivel en el instante anterior

DT es el intervalo de tiempo elegido

FE(JK) son los flujos de entrada en el período T

FS(JK) son los flujos de salida en el período T

Las *ecuaciones de flujo* tanto de entrada como de salida, se miden en intervalos de tiempo que van desde el instante actual al instante siguiente (KL):

$F(KL) = N(K) * FN * M(K)$ donde:

F es la variable de flujo

N es el nivel

FN es el flujo normal

M es un multiplicador del flujo normal (se usa para alterar el comportamiento normal de la variable).

6). *Calibrado.*

Lograda esta etapa se dice que el modelo queda precisado al máximo ya que se han establecido las formas de las funciones y los valores de los parámetros. Determinar el valor de un parámetro supone diseñar un experimento por parte de un experto.

7). *Análisis de sensibilidad.*

En esta etapa se estudia cómo varía las variables endógenas ante variaciones pequeñas de los parámetros. Cuando las trayectorias de una variable son semejantes se dice que es poco sensible al parámetro, caso contrario es muy sensible. Cuando se descubre que una variable es muy sensible a un parámetro hay que poner especial énfasis en el calibrado de éste.

8). *Evaluación del modelo: contrastado.*

Consiste en contrastar los valores que arroja el modelo respecto de los hechos observados. En función de ello se dice si el modelo es bueno o no (útil o no). La condición necesaria pero no suficiente para que un modelo sea bueno es que los resultados que arroje cumplan con una *norma de convergencia* establecida por el experto. Ésta está dada por $\frac{y^c - y^0}{y^0} * 100 \leq \varepsilon$, donde y^0 es el valor observado, y^c es el valor calculado y ε es un valor absoluto predeterminado.

9). *Utilización del modelo: escenarios e imágenes simuladas.*

La primera utilización del modelo es la apropiación del conocimiento del sistema que representa, y la simulación con el modelo puede proporcionar tal conocimiento. Para poder hacer la simulación se deben conocer: las *condiciones iniciales* (dadas por valores invariantes y objetivos de todos los niveles, de las variables de flujo, auxiliares y de los parámetros) y los *escenarios de simulación* (que son valores de variables que cambian y se establecen subjetivamente). Las *imágenes del sistema* son el conjunto de valores de las variables endógenas asociadas a cada conjunto de condiciones iniciales y escenarios de simulación.

II.4. *Marco Empírico*

Para este trabajo de investigación el universo de estudio consistirá en proyectos de desarrollo de software y como objeto de análisis se presenta un proyecto ya terminado (análisis post-mortem) que satisfaga, al menos, los criterios de registro de valores de:

* Tamaño
* Duración
* Mano de obra

En base a estos requisitos se elige como caso de estudio:

* Proyecto DE-A

Este proyecto fue utilizado para probar el modelo base de Abdel-Hamid y Madnick [1].

El objeto de emplear este caso de estudio es examinar la habilidad del modelo para reproducir la dinámica del proyecto software ya completado.

A partir de la documentación registrada para este proyecto se recogerán datos necesarios que serán utilizados como entrada para simular el entorno del proyecto mencionado, observar el comportamiento de la simulación y compararlo con el comportamiento real del proyecto.

Las salidas generadas para distintos escenarios de simulación serán comparadas con los valores reales ya conocidos del proyecto y de esta manera se podrá determinar si existe subestimación o sobrestimación y conocer los porcentajes de desviación entre los valores reales y estimados y cuan razonable son los resultados de la simulación.

CAPITULO III

.....

MODELO DE ABDEL-HAMID Y MADNICK

Presentación

La construcción del modelo de este trabajo de investigación esta basado en el primer modelo dinámico, el *"Modelo de Abdel-Hamid, Tarek y Madnick, Stuart"*, cuya elección se fundamenta por ser el modelo dinámico por excelencia, en relación a la gestión de proyectos de desarrollo de software, de referencia obligada para la creación de modelos dinámicos posteriores a él.

Como se indicó en el Capítulo II "*Marcos Referenciales*", el modelo tomado como base para el presente trabajo, se construye para hacer inferencias acerca del comportamiento del sistema como una totalidad socio-técnica desde el conocimiento de los microcomponentes tales como la planificación y productividad.

Sus dos características claves que lo distinguen de otros modelos [27] [31] [32] en el área de la Ingeniería del Software son las siguientes:

a).- Es *integrador*: en el sentido que integra las múltiples funciones del proceso de desarrollo de software, incluyendo tanto las funciones de *gestión* (planificación, control y asignación de personal) como las de *producción* de software que constituyen el ciclo de vida de desarrollo de software (diseño, codificación, revisión y prueba).

b).- Es un modelo de *Dinámica de Sistemas (DS)*: fundado en las distintas premisas de la filosofía de la DS, se establecieron dos bases principales para operacionalizar la técnica y sobre las que se apoya este modelo: una es el uso de los sistemas de feedback de información (para modelar y entender la estructura del sistema) y la otra es el uso de la simulación por computadora (lo que permite experimentar superando las barreras de la intuición y tiempo para estudiar y predecir implicaciones dinámicas).

Bajo estas características, es un modelo que usa los principios de feedback de la DS y la simulación para estructurar y clarificar la compleja red de variables que interactúan dinámicamente y que caracterizan la conducta de un sistema socio-técnico ya que integra las funciones tanto del nivel de gestión como del nivel de producción.

El foco de interés de este modelo está puesto en las fases de desarrollo de la producción de software, extendiéndose únicamente desde la fase de diseño hasta la última

fase de desarrollo llamada fase de prueba. Queda claro entonces que excluye las fases de análisis y mantenimiento.

III.1. *Estructura del Modelo*

La estructura del modelo consiste de cuatro subsistemas mayores, *gestión de recursos humanos, planificación, control y producción de software,* junto con los diferentes flujos que los conectan. (Ver Figura II.1. – Capítulo II)

Estos subsistemas se describen detalladamente en las secciones siguientes, cada uno con sus diferentes sectores y siguiendo las convenciones esquemáticas que se usan en la metodología Dinámica de Sistemas.

Las variables que unen los subsistemas están resaltadas en color azul.

III.1.1. *Subsistema "Gestión de Recursos Humanos"*

Este subsistema incluye la contratación, despido, entrenamiento (formación), asimilación, rotación y retiro de los recursos humanos de un proyecto. Su modelo se muestra en la Figura III.1.

En el modelo del subsistema se asume que el "*Total de mano de obra*" del proyecto consiste de dos niveles de mano de obra: "*Mano de obra recientemente contratada*" y "*Mano de obra experimentada*". La distinción en estas dos categorías de mano de obra tiene dos razones:

a).- Los miembros nuevos pasan por un período de *orientación* sobre el proyecto y la organización, durante el cual son menos productivos. La orientación abarca tanto dimensiones técnicas (metodología del proyecto, técnicas de programación, hardware, software) como sociales (normas y reglas organizacionales, relación con otros miembros, a quien consultar ante determinados problemas, etc.).

b).- El *entrenamiento* de los miembros recién contratados lo realizan los miembros experimentados. Esto implica que mientras un empleado experimentado está ayudando al empleado nuevo, se reduce la propia productividad sobre su otro trabajo.

Se aclara que no todos los miembros del proyecto son reclutados desde afuera de la organización, sino que son *trasferidos* desde otros proyectos de la misma organización. Pero incluso en este caso, el empleado transferido tendrá su período de orientación para aprender reglas bases del proyecto, objetivos del esfuerzo, plan de trabajo y todos los detalles del sistema. Aunque obviamente será menos costosa que la orientación necesaria para un reclutado fuera de la organización.

Determinar cuánto esfuerzo comprometer para el *entrenamiento* de nuevos empleados, es generalmente un resultado intuitivo de la gestión. En el modelo se estableció como un parámetro: "*Entrenador por contratado reciente*" y su valor se estableció en 0,20. Esto significa que, en promedio, cada nuevo empleado consume en entrenamiento un 20% del tiempo de un empleado experimentado.

El período promedio de *asimilación* se formula como un retraso exponencial de 1° orden. En el modelo este retraso se estableció en 80 días. Es decir que después de transcurridos 80 días de la contratación, el empleado nuevo pasará a ser un empleado experimentado.

Para decidir el nivel de "*Total de mano de obra*" la gestión tiene en cuenta muchos factores.

- *Fecha de compleción programada actual*: se determina la mano de obra que se cree necesaria para completar las tareas percibidas restantes, dentro de esa fecha.
- *Estabilidad de la mano de obra*: se estima por cuanto tiempo se necesitarán los empleados nuevos.
- *Habilidad para absorber nuevos empleados*: se restringe la contratación de nuevos empleados a un número que, se estima, es aquel que un empleado experimentado full-time puede manejar eficientemente. En el modelo se fijó en 3 empleados.

Por todo esto es que se determina un "*Tope de nuevos contratos*" que estará determinado por el producto entre el máximo número de nuevos empleados que un miembro experimentado puede manejar y el número de empleados experimentados full-time (éste último no siempre es igual al nivel "*Mano de obra experimentada*" ya que un empleado experimentado puede estar asignado a más de un proyecto).

La suma de "*Tope de nuevos contratos*" y "*Mano de obra experimentada*" determinan el "*Tope de total de mano de obra*". El valor de esta variable representa el

máximo número de empleados que la gestión desea contratar y se usa para limitar la "*Mano de obra buscada*" según la "*Mano de obra necesaria*".

Una vez determinada la "*Mano de obra buscada*" pueden ocurrir tres situaciones que la gestión deberá enfrentar:

1.- El "*Hueco de mano de obra*" entre "*Mano de obra buscada*" y "*Total de mano de obra*" es 0 (cuando los niveles son iguales). En este caso no es necesaria acción alguna.

2.- "*Mano de obra buscada*" es mayor que "*Total de mano de obra*". Es el caso más probable y deben ser contratados nuevos empleados. Esto por supuesto lleva tiempo. El retraso en contratar nuevos empleados se establece en 40 días.

3.- "*Mano de obra buscada*" es menor que "*Total de mano de obra*", siendo la acción necesaria transferir miembros fuera del proyecto. Si hay miembros en entrenamiento, éstos serán los primeros en ser transferidos. Si todavía es necesario realizar más transferencias, se hacen con los empleados experimentados. Los empleados transferidos deben realizar algún trabajo (documentación por ejemplo) antes de retirarse, por lo cual se establece un retraso de transferencia de 10 días.

Finalmente, hay un efecto sobre el proyecto por el retiro de la mano de obra. En el modelo se estableció una "*Tasa de retiro*" de 30% pero sólo para "*Mano de obra experimentada*" ya que se asume que es improbable que un empleado nuevo se retire dentro de los 80 días (tiempo que toma para convertirse en empleado experimentado) de unirse al proyecto. A partir de esta tasa se puede determinar el "*Tiempo promedio de empleo*" (como retraso exponencial de 1° orden) de 673 días.

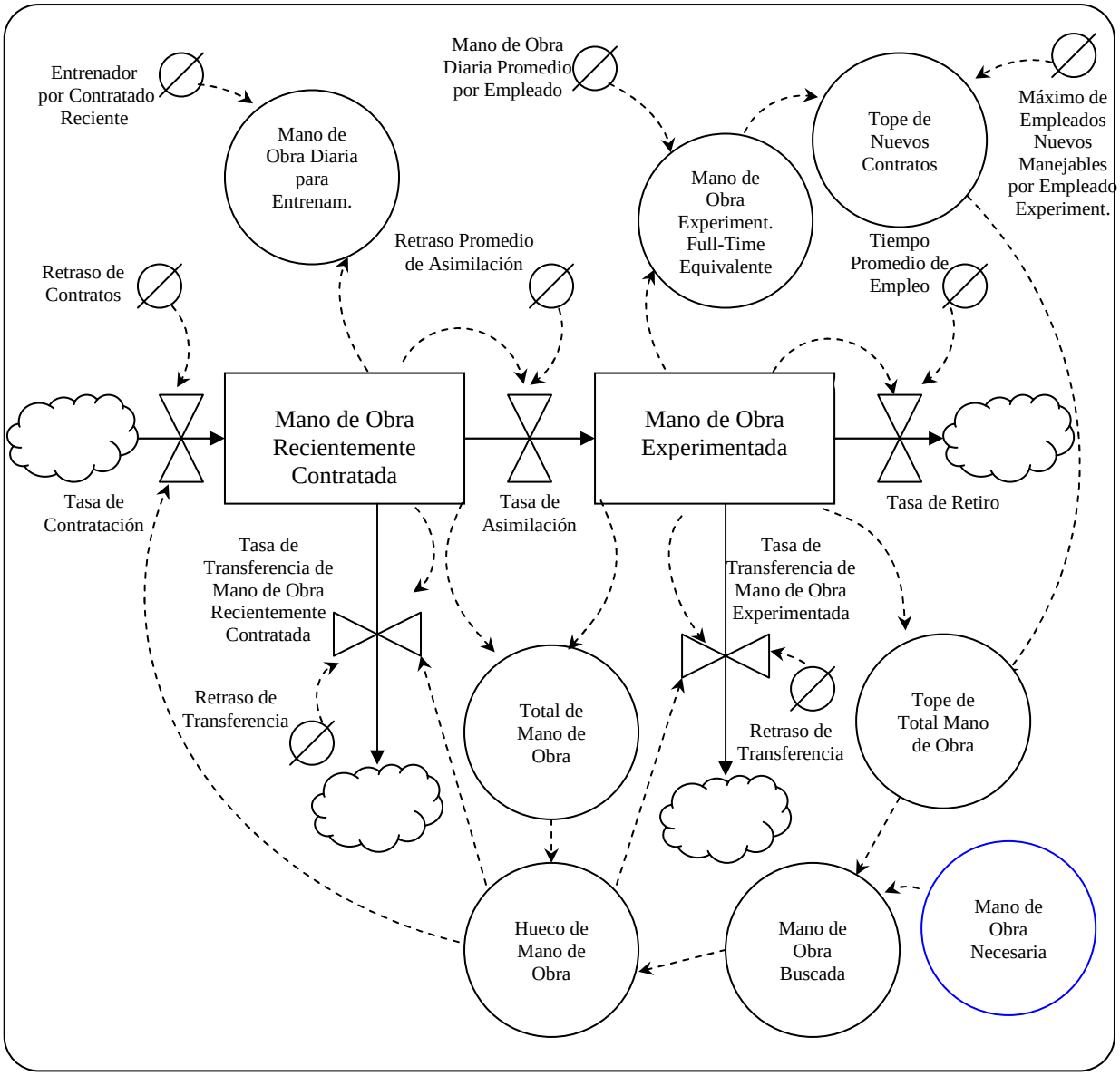


Figura III.1. Subsistema "Gestión de Recursos Humanos"

III.1.2. *Subsistema "Producción"*

La producción de software es la mayor actividad de un proyecto de desarrollo de software. En el modelo hay cuatro actividades primarias dentro del subsistema de producción de software. La actividad de desarrollo incluye tanto el *diseño* como la *codificación* del software. Mientras el software se está desarrollando, también se *revisa* para detectar cualquier error de diseño o código. Luego los errores detectados a través de las actividades de Garantía de Calidad (en inglés Quality Assurance - QA) se *corrigen*

(rework). Sin embargo, no todos los errores serán detectados durante la fase de desarrollo, algunos pasarán inadvertidos hasta la fase de *prueba*.

Dada la complejidad de este subsistema para explicarlo como una sola pieza, se lo modela dividiéndolo en los cuatro sectores siguientes:

1. Asignación de mano de obra
2. Desarrollo de software (incluye diseño y codificación)
3. Garantía de calidad y rework
4. Prueba

III.1.2.1. *Sector "Asignación de Mano de Obra"*

Como se indica en la Figura III.2. que corresponde a este sector, el "*Total de mano de obra diaria*" es una función (producto) de "*Total de mano de obra*" y "*Mano de obra diaria promedio por empleado*". Ésta última tendrá valores diferentes para los casos en que un empelado sea asignado a un solo proyecto (valor 1) o a dos proyectos (valor 0,5).

Parte de la mano de obra disponible, será consumida en el entrenamiento. Esta parte es la que después del entrenamiento se asigna para QA, rework, desarrollo y prueba.

Del "*Total de mano de obra diaria*" una parte se asigna a las actividades de QA. Esa parte, "*Fracción planeada de mano de obra para QA*" se estableció a un nivel uniforme de 15% y como una función de "*% de trabajo realizado*" para poder experimentar con otras políticas de QA, donde el esfuerzo de QA no se distribuye uniformemente a lo largo del ciclo de vida.

La "*Fracción real de mano de obra para QA*" puede ser diferente de "*Fracción planeada de mano de obra para QA*" esto debido a la "*Presión de tiempo*", donde:

$$\textit{Presión de tiempo} = \frac{\textit{(Esfuerzo total percibido todavía necesario para completar el proyecto – Esfuerzo total restante en el plan actual)}}{\textit{Esfuerzo total restante en el plan actual}}$$

- Si *Esfuerzo total percibido todavía necesario para completar el proyecto* es igual a *Esfuerzo total restante en el plan actual* la presión es nula.

- Si *Esfuerzo total percibido todavía necesario para completar el proyecto* es mayor que *Esfuerzo total restante en el plan actual* la presión es positiva y resultará, probablemente en trabajo extra, o más contratación o ambas cosas.

- Si *Esfuerzo total percibido necesario todavía para completar el proyecto* es menor que *Esfuerzo total restante en el plan actual* la presión es negativa y resultará en ocio (fracción de tiempo de proyecto gastado en actividades no relacionadas al proyecto).

Se pueden hacer estimaciones de la relación entre "*Presión de tiempo*" y esfuerzo de QA. En ausencia de presión ("*Presión de tiempo*" igual a 0), el porcentaje de ajuste de la "*Fracción planeada de mano de obra para QA*" será también 0, es decir, el esfuerzo de real de QA será igual al esfuerzo planeado. Cuando la presión de tiempo aumenta, el esfuerzo de QA disminuye. Esto es, las actividades de QA se relajan y se hacen "recortes" (alisado) en el esfuerzo planeado de QA, aunque no se eliminan completamente.

Ya se indicó que de la mano de obra disponible se consumió en entrenamiento y actividades de QA. El volumen restante de mano de obra llamada "*Mano de obra diaria para producción de software*" es asignado a las actividades de desarrollo de software (diseño y código), prueba y rework.

Cuando los errores se detectan a través de las actividades de QA, se asigna mano de obra para corregirlos (rework). La cantidad de esfuerzo diario asignado para ello es una función de "*Tasa de corrección de error deseada*" y "*Mano de obra de rework percibida necesaria por error*". En otras palabras, el esfuerzo se asigna basado en el trabajo de rework que será realizado y la productividad de rework percibida.

"*Mano de obra de rework percibida necesaria por error*", en el modelo, está diagramada como un tipo especial de nivel, en el cual su entrada no es una tasa. Esta es una "notación tipográfica" de una operación de alisado exponencial. Es decir "*Mano de obra de rework percibida necesaria por error*" es el "alisado exponencial" de su entrada: "*Mano de obra real de rework necesaria por error"*.

La otra variable que define "*Mano de obra diaria para rework*" es "*Tasa de corrección de error deseada*" esta es, la tasa diaria a la cual serán corregidos esos errores

descubiertos. Esta resulta de dividir el número de errores descubiertos por un "*Retraso de rework deseado*". El valor para este último se fija en 15 días, que sería el tiempo que trascurre para que un error sea corregido.

Después de asignar mano de obra a las actividades de rework, la porción restante de "*Mano de obra diaria para producción de software*" se dedica al desarrollo (diseño y codificación) y actividades de prueba.

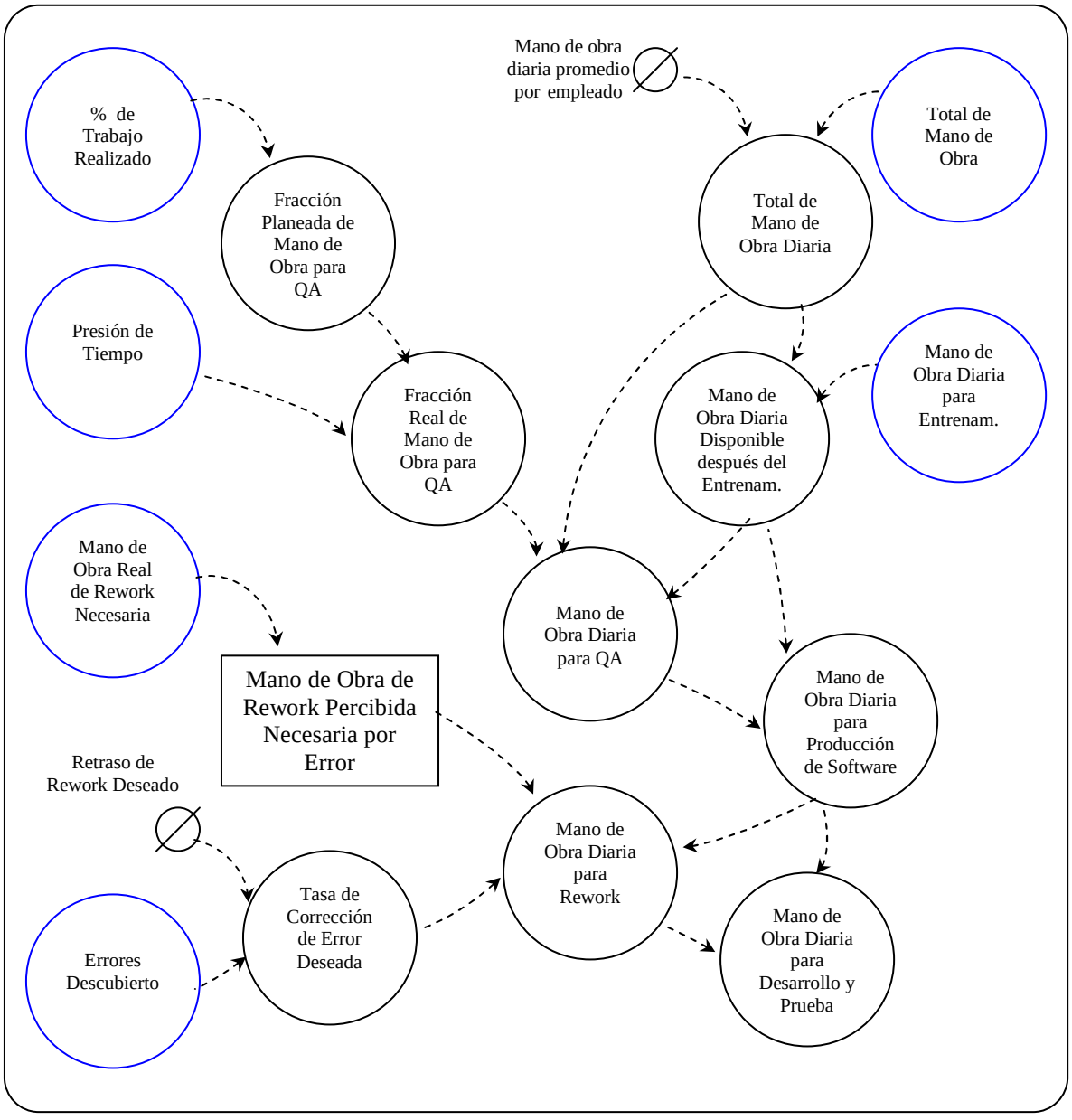


Figura III.2. Sector "Asignación de Mano de Obra"

III.1.2.2. *Sector "Desarrollo de Software"*

El proceso de desarrollo de software consiste en el diseño y codificación del producto software. Un proyecto software se define como un número de "tareas". Así, la tasa de desarrollo de software es una función de "*Tareas por día*", software desarrollado de "*Tareas desarrolladas*" y productividad de desarrollo de "*Tareas por día-hombre*". El modelo del sector de desarrollo de software se muestra en la Figura III.3.

La asignación de mano de obra al desarrollo, continua hasta que se percibe que la mayoría de las tareas de desarrollo de software se han completado, punto en el cual, comienza la fase de prueba del sistema y la mano de obra se asigna a la prueba. Este cambio en la asignación de la mano de obra es cuantificado en el modelo a través de la variable "*Fracción de esfuerzo para la prueba del sistema*". El valor de ésta es inicializado en 0, y se vuelve 1 cuando se percibe que se han completado todas las tareas de desarrollo. Que sea 1 significa que el 100% del esfuerzo disponible para desarrollo y prueba se usa en actividades de prueba del sistema. Este cambio no es abrupto, sino que hay un *solapamiento* entre las fases de desarrollo y prueba. Esto es, gradualmente aumenta el valor de "*Fracción de esfuerzo para la prueba del sistema*" mientras la fracción de tareas de desarrollo percibidas restantes decrece.

Durante la fase de desarrollo, la tasa a la cual se desarrolla el software es una función no sólo de cuanta mano de obra se usa, sino también de cuan productivos son quienes desarrollan el software.

La "*Productividad de desarrollo de software*" es una función de un complejo conjunto de factores. Según Ivan Steiner:

Productividad Real = Productividad Potencial – Pérdidas debido a procesos defectuosos

donde

- *Pérdidas debido a procesos defectuosos* se refiere básicamente a las pérdidas por comunicación y motivación.

- *Productividad potencial* se define como el máximo nivel de productividad que puede ocurrir cuando un individuo o grupo emplea sus recursos para lograr la tarea demandada por una situación de trabajo. Esto se logra cuando el individuo o grupo hace el mejor uso posible de los recursos, es decir, cuando no hay pérdidas por procesos defectuosos.

- *Productividad real* que logra un individuo raramente iguala a la productividad potencial ya que usualmente no hace el mejor uso posible de los recursos disponibles. Problemas de coordinación y/o motivación son responsables de insuficiencias en los procesos y de consiguientes pérdidas de productividad.

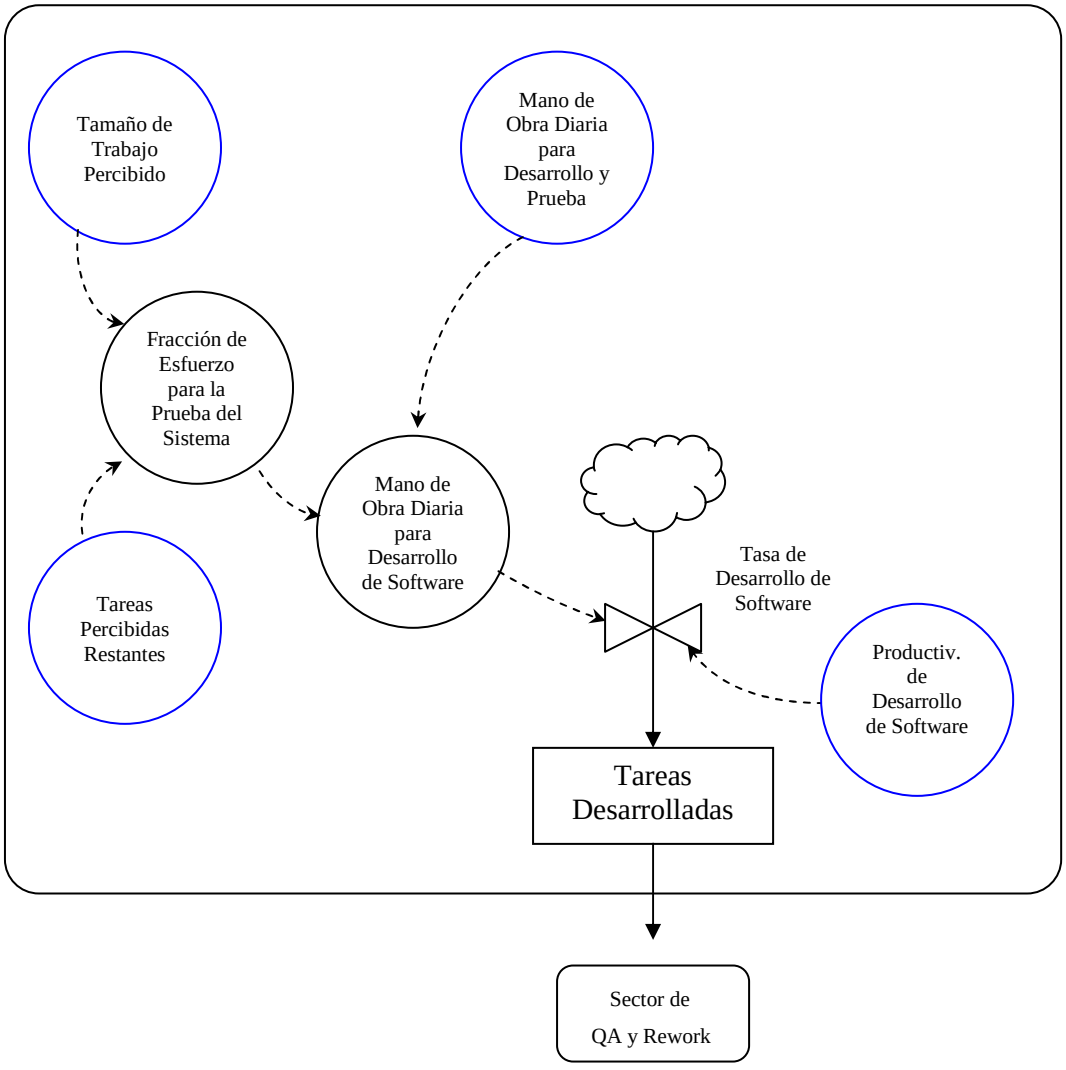


Figura III.3. Sector "Desarrollo de Software"

III.1.2.2.1. *Subsector "Productividad de Desarrollo de Software"*

La "*Productividad potencial normal*" representa el máximo nivel de productividad de desarrollo de software que puede ocurrir cuando un individuo emplea los recursos disponibles para realizar las tareas de un proyecto.

Este parámetro está definido en términos de un número de "tareas / hombres/día" lo cual significa que su valor depende de cómo se define una tarea. En el modelo de productividad, Figura III.4., se define "*Productividad potencial normal*" como un cierto número de "tareas / hombres/día" donde "tarea" es una unidad de medida del tamaño del producto software (tal como LOC, PF, etc.). En particular en el modelo una tarea se define en términos de un número de Instrucciones Fuente Entregadas (IFE – DSI en inglés).

La "*Productividad potencial normal*" se ve afectada por dos factores que tienen la particularidad de no mantenerse constantes a lo largo de un proyecto particular:

- Nivel de experiencia de la mano de obra
- Aumento de la familiaridad con el proyecto debido a la curva de aprendizaje

Para integrar al modelo el efecto de la experiencia, se usan dos parámetros: uno para representar la productividad potencial normal de los empleados experimentados y otra para representar lo mismo pero para los empleados recientemente contratados.

Cada uno de estos dos parámetros se pesan por la fracción de sus correspondientes tipos de empleados en el total de mano de obra y determina la "*Productividad potencial normal promedio*" que cambia cuando cambia la mezcla de empleados nuevos y experimentados. En el modelo los valores de estos dos parámetros se establecen en 1 y 0,5 respectivamente.

El segundo factor que afecta la productividad potencial es el incremento de la habilidad debido a la curva de aprendizaje, esto porque a medida que avanza un proyecto, los empleados aprenden mejor su trabajo,

por lo que la curva de aprendizaje es la tasa de mejora. En el modelo el efecto de la curva de aprendizaje se formula como un "*Multiplicador de productividad debido al aprendizaje*" que tomará valor 1 (cuando se está al comienzo del proyecto) y aumentará (hacia el final del proyecto) indicando la "tasa de mejora" en la productividad.

La relación de la "*Productividad real*" y "*Productividad potencial*" con las pérdidas debido a la comunicación y motivación se indica en el modelo haciendo "*Productividad de desarrollo de software*" igual al producto entre "*Productividad potencial*" y el "*Multiplicador de la productividad debido a pérdidas por comunicación y motivación*". En ausencia de estas pérdidas este multiplicador asume el valor 1 y la "*Productividad potencial*" es igual a la "*Productividad real*". Si existen pérdidas el multiplicador asumirá valores menores que 1, conduciendo el valor de la productividad real a valores por debajo de la productividad potencial.

Los efectos de la motivación y comunicación son multiplicativos:

a).- Los factores de motivación determinan la fracción de esfuerzo dedicada a actividades no relacionadas proyecto. Esta fracción usualmente es menor que 1 dado que con frecuencia hay pérdidas de tiempo en actividades como lectura de mails, cafés, cuestiones personales, etc.

b).- Los factores de comunicación determinan la fracción de horas de proyecto. Se refiere a pérdidas de comunicación del tipo proyecto.

Las pérdidas por factores de motivación están relacionadas con un "tiempo de ocio", esto es, la fracción de tiempo de proyecto gastada en actividades no relacionadas al proyecto. Por ello las pérdidas de motivación son formuladas en términos de pérdidas de horas-hombre.

En el modelo el mecanismo de motivación está diseñado para capturar el impacto motivacional de la presión de tiempo sobre el "tiempo

de ocio" de los miembros del proyecto. Es decir el rol motivacional de la presión de tiempo puede ser expandir o contraer el tiempo de ocio.

En ausencia de presión, la fracción de horas diarias asignadas al trabajo relacionado al proyecto, por un miembro full-time del equipo, es definida por el parámetro "*Fracción normal de esfuerzo sobre el proyecto*". En el modelo este parámetro se estableció a en 60%. Esto significa que en ausencia de presión un empleado full-time dedica en promedio 0,6 * 8 = 4,8 hs. al proyecto (asumiendo un día de trabajo de 8 hs.). Entonces, bajo condiciones normales, la contribución de las pérdidas por motivación al "*Multiplicador de la productividad debido a pérdidas por comunicación y motivación*" lleva a un recorte del 40% en la productividad potencial. Los efectos de la motivación de presión pueden llevar a valores más altos o más bajos de "*Fracción real de esfuerzo sobre el proyecto*", que inicialmente es igual al valor de "*Fracción normal de esfuerzo sobre el proyecto*", valor que se mantiene mientras no haya presión.

Si hay presión positiva, la diferencia del numerador de su fórmula representa la escasez percibida en esfuerzo. En este caso se trabaja extra o se asigna más esfuerzo al proyecto para compensar la escasez. Es decir se comprime el "tiempo de ocio" aumentando el esfuerzo diario dedicado al proyecto.

Trabajar horas extras, eliminando el "tiempo de ocio", llevaría en algún momento al que se produzca un agotamiento en los empleados, es decir que existe un *umbral*.

Cuando la diferencia "*Esfuerzo total percibido todavía necesario para completar el proyecto*" menos "*Esfuerzo total restante en el plan actual*" (esta diferencia corresponde al numerador de la ecuación que define la presión de tiempo) es positiva (proyecto retrasado), dos factores determinan el nivel al cual se "ajusta" la "*Fracción real de esfuerzo sobre el proyecto*":

- Si la diferencia que es "*Escasez percibida en esfuerzo*", está por debajo del umbral, los empleados ajustarán las horas de trabajo asignadas al proyecto, al total de horas percibidas como escasas.
- Existe una "*Máxima escasez de esfuerzo a ser manejado*" y constituye el umbral mencionado. Si la escasez percibida es

mayor que el máximo, los empleados estarán deseosos (se los motiva) de trabajar extra para acomodarlo al valor máximo y la gestión decide extender el plazo para manejar lo que excede a "*Máxima escasez de esfuerzo a ser manejado*".

Pero la tolerancia para trabajar a ese ritmo decrece y el valor de "*Máxima escasez de esfuerzo a ser manejado*" decrece.

El valor de "*Máxima escasez de esfuerzo a ser manejado*" está determinado por el producto de tres variables: "*Umbral de duración de sobrecarga*", "*Mano de obra full-time equivalente*" y "*Máximo ajuste en horas/hombre*". El valor de éste último se fijó en 100%. El valor fijado para "*Umbral de duración de sobrecarga*" es 50 días de trabajo (10 semanas). Una vez que se empieza a trabajar duro, este umbral (que representa la duración máxima restante para trabajar duro) decrece por debajo del valor normal. En el modelo este decrecimiento se realiza por efecto de un multiplicador, "*Multiplicador del umbral de duración de sobrecarga debido al cansancio*".

Cansancio es un nivel cuyo valor refleja el nivel de agotamiento de la mano de obra debido a la sobrecarga. La tasa a la cual crece ese nivel está dado por "*Fracción normal*" y "*Fracción real*" de esfuerzo sobre el proyecto. Si la fracción real es menor o igual que la fracción normal, el valor de la "*Tasa de incremento del nivel de cansancio*" es 0 ya que como los empleados están trabajando a un paso normal, no hay aumento en su nivel de cansancio. La "*Tasa de cansancio*" es una función de la compresión del tiempo de ocio.

Como se mencionó anteriormente el efecto del cansancio sobre el "*Umbral de duración de sobrecarga*" es formulado por el "*Multiplicador del umbral de duración de la sobrecarga debido al cansancio*". El valor inicial de ese umbral es 50 días y cuando el empleado comienza a trabajar a una tasa superior a la normal ese umbral disminuirá hasta alcanzar el valor 0.

Una vez que el período de sobrecarga finaliza, ya sea porque el umbral ha sido alcanzado y/o porque la presión cesa, la mano de obra retorna a su tasa de trabajo normal. La "*Tasa de vaciamiento del nivel de*

cansancio" se modela como un retraso exponencial de primer orden con un tiempo de retraso de 4 semanas.

Determinar el valor de "*Umbral de duración de sobrecarga*" es necesario para determinar el valor de "*Máxima escasez de esfuerzo a ser manejado*", cuyo valor a su vez es necesario para determinar el valor al cual se ajusta la "*Fracción real de esfuerzo sobre el proyecto*". Cuando hay presión positiva (proyecto atrasado) los empleados ajustarán su tasa de trabajo ya sea a "*Escasez percibida en esfuerzo*" o "*Máxima escasez de esfuerzo a ser manejado*". El más pequeño de estos valores constituye el "*Esfuerzo manejado*". El "*Porcentaje de ajuste en la tasa de trabajo buscado*" para manejar ese esfuerzo, iguala al valor de "*Esfuerzo manejado*" dividido por el producto de "*Mano de obra equivalente full-time*" y "*Umbral de duración de sobrecarga*".

El "*Porcentaje de ajuste en la tasa de trabajo buscado*" define una meta de tasa de trabajo en términos de esfuerzo asignado al proyecto. Una meta tal no se alcanza de forma instantánea, dado que a los empleados les toma un tiempo ajustar sus hábitos de trabajo. Por eso hay un retraso antes que "*Fracción real de esfuerzo sobre el proyecto*" alcance el nivel buscado. Ese retraso promedio se fijó en 2 semanas.

Si la diferencia "*Esfuerzo total percibido todavía necesario para completar el proyecto*" menos "*Esfuerzo total restante en el plan actual*" es negativa (proyecto adelantado) los excesos son traducidos en cortes en los tiempos programados del proyecto. Absorber estos excesos significa mayor "tiempo de ocio" lo que a su vez significa una más baja "*Fracción real de esfuerzo sobre el proyecto*". Esta fracción se determinará (como en el caso de presión positiva) a través de un ajuste de la variable "*Porcentaje de ajuste en la tasa de trabajo buscado*" pero en este caso, el porcentaje de ajuste será un valor negativo. Es decir que hay un ajuste a una estable pero cómoda baja tasa de trabajo. El tiempo de retraso para hacer ese ajuste es de 7,5 días. Es 25% más bajo que en el caso del retraso de ajuste cuando hay presión positiva, porque es más rápido ajustar los hábitos de trabajo a estados más confortables.

Ya se indicó que "*Productividad de desarrollo de software*" es igual al producto de "*Productividad potencial*" y "*Multiplicador de productividad debido a las pérdidas por motivación y comunicación*". El multiplicador representa la fracción productiva promedio de un empleado por unidad de tiempo: la fracción de la "*Fracción real de esfuerzo sobre el proyecto*" que permanece después de considerar la sobrecarga de comunicación.

La sobrecarga de comunicación es la caída promedio de la productividad de un miembro del equipo por debajo de la productividad normal como resultado de la comunicación (verbal, documentación, interfaces y cualquier otro trabajo adicional). Según estudios se sabe que la sobrecarga de comunicación incrementa en una proporción n^2, donde n es el tamaño del equipo.

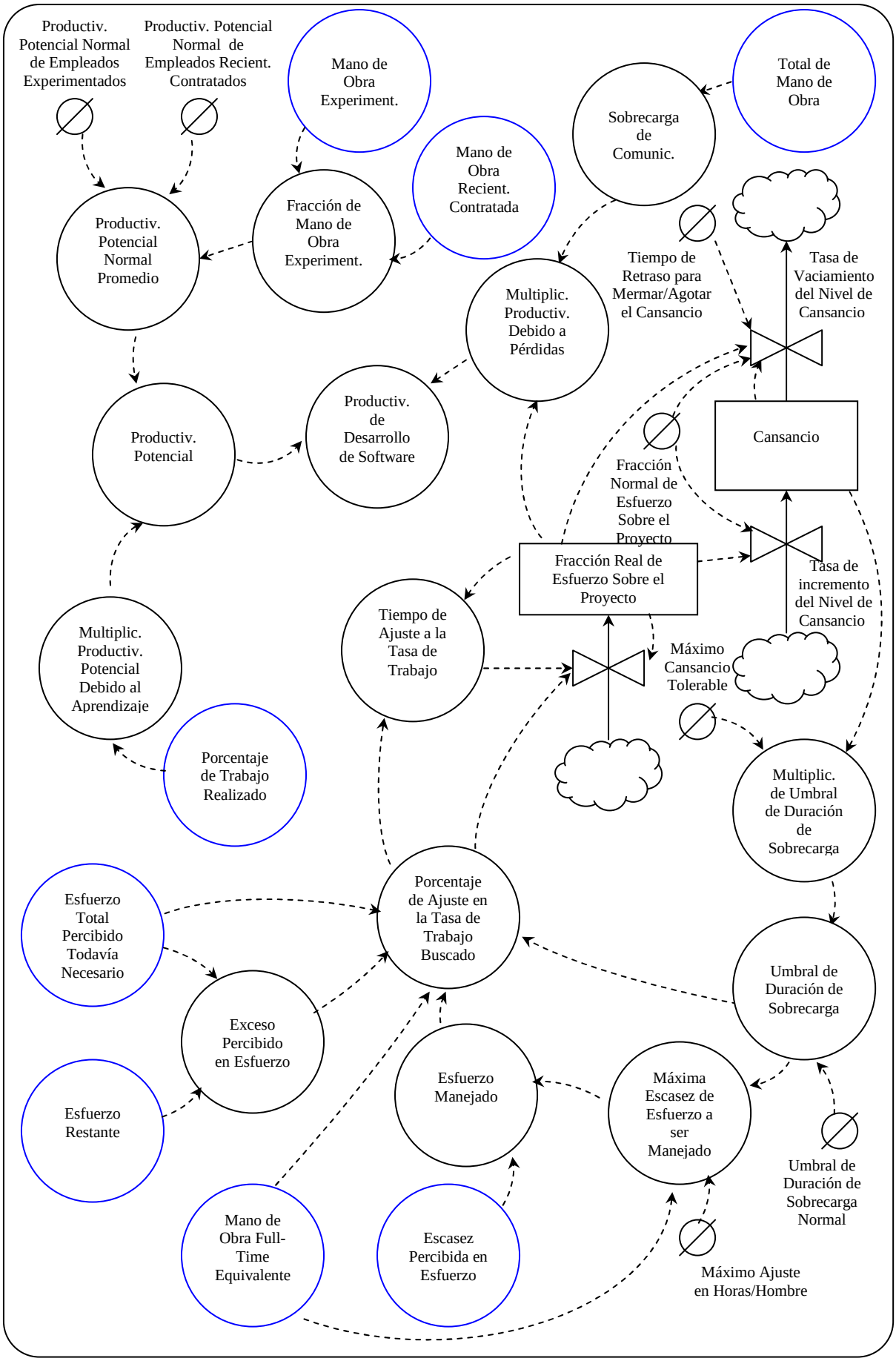


Figura III.4. Subsector “Productividad de Desarrollo de Software”

III.1.2.3. *Sector "Garantía de Calidad" (QA) y "Rework"*

Dada la dificultad humana de realizar con perfección el proceso de desarrollo de software, éste es acompañado de la actividad de Garantía de Calidad (QA). Errores pueden ocurrir al principio del proceso donde los objetivos del sistema software pueden ser especificados errónea o deficientemente, así como durante el posterior diseño y desarrollo donde ya se mecanizan estos objetivos.

La QA de software incluye dos metodologías distintivas y complementarias. La primera es el diseño de un coherente, completo, no ambiguo y no conflictivo conjunto de requisitos (no incluida en este modelo base por asumir que el diseño comienza con una revisión completa de requisitos de software que no sufrirán sucesivas modificaciones). La segunda es la revisión y prueba del producto.

La Figura III.5. ilustra el sector de QA y Rework del modelo.

La "*Tasa de generación de error*" está afectada por dos conjuntos de factores:

- Factores organizacionales: el uso de técnicas estructuradas, la calidad de los empleados.
- Factores de proyecto: complejidad, tamaño del sistema, lenguaje empleado.

Aunque estos factores pueden variar de organización en organización y de proyecto en proyecto, permanecen estables durante la vida de un mismo proyecto. El efecto acumulativo de estos factores se representa en el modelo a través de la variable "*Errores normales cometidos por tarea*". La tasa normal de generación de error es el producto de "*Tasa de desarrollo de software*" (cuántas tareas son desarrolladas por unidad de tiempo) y "*Número normal de errores cometidos por tarea*". Dado que esta simple variable representa diferentes tipos de errores, no se formula como una constante sino como una variable que cambia a lo largo de la vida del proyecto.

La formulación de "*Errores normales cometidos por tarea*" esta definida en KSDI en lugar de tareas, expresiones que son equivalentes ya que una tarea en sí misma está definida en términos de DSI.

Resultados encontrados en la literatura, indican que los errores son generados a diferentes tasas en diferentes etapas del ciclo de vida. Por ejemplo, los errores de diseño son generados a una tasa mayor que los errores de código.

En el modelo el ratio alcanza un valor máximo de 2:1. Esto es, al comienzo del diseño, el número normal de errores cometidos es 25 errores/KDSI, mientras que hacia el final del código cae a 12,5 errores/KDSI. Las tasas promedio para las fases de diseño y código son aproximadamente 23 y 14,5 errores/KDSI respectivamente, un ratio de 1,6:1.

Un segundo conjunto de factores que, a diferencia del primer conjunto de factores los cuales permanecen constantes durante la vida del proyecto, juegan una rol dinámico durante el desarrollo de software son:

a).- La mezcla de mano de obra

b).- La presión de tiempo

En relación al primero, ya se indicó que los empleados recién contratados son menos productivos pero además son más propensos a cometer errores. En el modelo se asume que un empleado nuevo es 2 veces más propenso a cometer errores que un empleado experimentado. Para modelar esto se formula el "*Multiplicador de generación de error debido a la mezcla de mano de obra*" como una función del "*Porcentaje de mano de obra experimentada*". Cuando la mano de obra consiste sólo de empleados experimentados, el valor del multiplicador se establece en 1, dado que tendría un efecto neutral sobre la tasa normal de generación de error. Cuando la fracción de empleados nuevos cree, el multiplicador incrementa linealmente hasta lograr el valor máximo de 2, lo cual significa que la mano de obra consiste únicamente de nuevos contratados.

La presión de tiempo, el segundo factor, también incrementa el número de errores. Bajo presión no se trabaja mejor sino más rápido. Cuando hay presión las actividades se solapan, cuando se hubieran cumplido mejor si se las realizara secuencialmente y esto incrementa significativamente la posibilidad de error.

En el modelo, bajo condiciones normales no hay presión y el "*Multiplicador de generación de error debido a presión de tiempo*" es 1. Cuando la presión aumenta el multiplicador aumenta exponencialmente, indicando tasas de generación de error superiores. Bajo condiciones de presión negativa, los errores serán generados a una tasa debajo de lo normal.

Los errores son "*Errores potencialmente detectables*" hasta que una tarea se revisa y se prueba, punto en el cual algunos errores serán descubiertos y corregidos (rework).

Algunos errores pasan inadvertidos a la siguiente fase (etapa de prueba) donde serán descubiertos y corregidos, aunque a un costo mayor.

La detección de errores es el objetivo de la actividad de QA. La "*Tasa de QA*" está formulada como un retraso de tercer orden. La forma característica para formular una tasa es como producto del esfuerzo asignado y su productividad. Sin embargo, esta tasa de QA es independiente del esfuerzo de QA y su productividad. Esto es así porque las actividades de QA, independientemente de cuantas tareas necesiten ser procesadas, ellas siempre serán probadas. Incluso las actividades de QA pueden relajarse o suspenderse debido a la presión, pero no retrasa el desarrollo. Y esto porque como es imposible decir si todos los errores fueron detectados, también es difícil decir que el trabajo de QA se realizó completamente. Además el esfuerzo que es posible gastar en QA es lo que se gasta y no más, porque no hay ningún incentivo significativo para hacerlo de otra manera.

Entonces las tareas de software siempre serán revisadas después de un retraso que se asume es independiente del esfuerzo de QA. En el modelo ese retraso se estableció en 2 semanas (10 días de trabajo). Aunque la tasa a la cual las tareas se revisan o se consideradas revisadas es independiente del esfuerzo real asignado a QA, la efectividad de QA depende del esfuerzo. Esto es, el número de errores detectados estará en función de cuanto esfuerzo de QA se asigna a la detección de error.

En el modelo la variable "*Tasa de detección de error potencial*" representa el máximo número de errores que pueden ser detectados a la vez y está determinado por el cociente entre el valor de esfuerzo de QA asignado y el valor del esfuerzo de QA necesario, en promedio, para detectar un error.

Determinar la "*Mano de obra de QA necesaria para detectar un error*" implica detectar si el error es de diseño o de código.

La mano de obra de QA *real necesaria* para detectar un error es una función del tipo de error y de cuan eficientemente trabajan los empleados. Por eso su formulación depende del "*Multiplicador de productividad debido a las pérdidas por motivación y comunicación*". Si hay tales pérdidas, la mano de obra para QA *real necesaria* para detectar un error será mayor que la mano de obra *normal necesaria.*

A pesar que se realicen asignaciones generosas de esfuerzo de QA, quedarán errores sin detectar hasta la integración del sistema. Por ello el nivel "*Errores potencialmente detectables*" se puede ver como una jerarquía de errores, donde algunos son más sutiles y por lo tanto más costosos de detectar que otros. En el modelo se asume que las actividades de QA detectarán primero los errores más obvios. Esto vuelve más costoso de detectar los errores restantes. Por eso en el modelo se usa un "*Multiplicador de esfuerzo de detección debido a la densidad de error*" que asume el valor neutral 1 para densidades de error de grande a moderado. Pero como todos los errores más obvios serán detectados y unos pocos quedarán sin detectar, el multiplicador aumenta exponencialmente indicando mayor costo para detectar un error.

Los errores detectados a través de QA se corrigen. La tasa de rework es una función de cuanto esfuerzo es asignado a las actividades de rework y la mano de obra de rework necesaria por error.

Al igual que en la detección de error, la "*Mano de obra de rework real necesaria por error*" está en función de "*Mano de obra de rework normal necesaria por error*" y del tipo de error (de diseño o de código). También, como en el caso de la detección, los errores de diseño son más costosos de corregir. Según resultados empíricos el esfuerzo de rework promedio está en el rango de 0,25 a 1,0 días-hombre por error.

La "*Mano de obra de rework real necesaria por error*" además de depender del tipo de error, también depende de cuan eficientemente trabajan los empleados. Como en la detección, si hay pérdidas por motivación y comunicación, el "*Multiplicador de productividad debido a pérdidas por motivación y comunicación*" indicará cuantas veces superará la "*Mano de obra de rework real necesaria*" a la "*Mano de obra de rework normal necesaria*".

Las correcciones llevan a arreglos malos, pero no hay datos que indiquen cual es la fracción de esas correcciones malas. Sin embargo algunos resultados indican que los arreglos malos constituyen entre el 6,5 y 10% de todos los errores corregidos en la etapa de prueba del sistema. En el modelo se estableció el "*Porcentaje de arreglos malos*" en 7,5%.

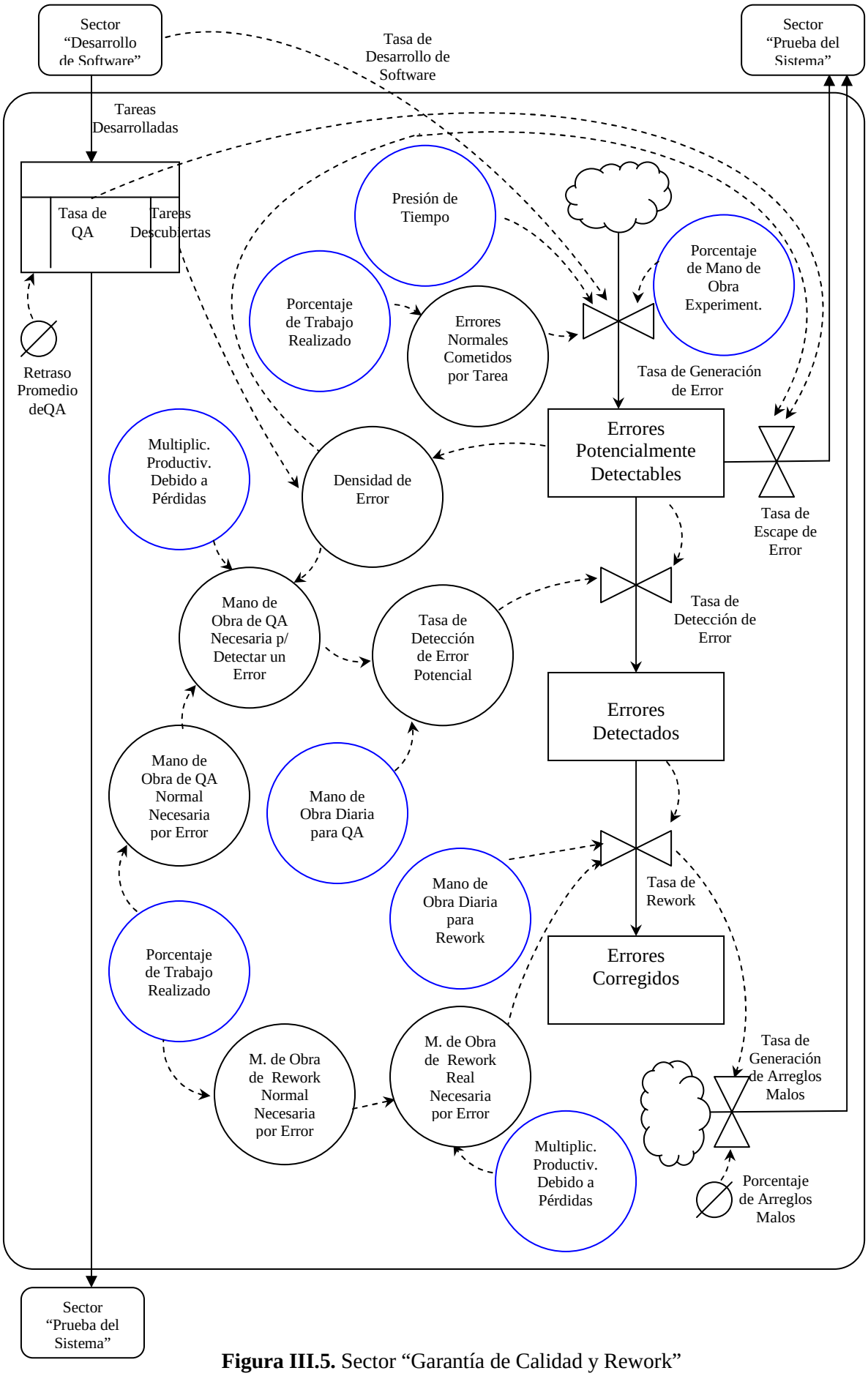


Figura III.5. Sector "Garantía de Calidad y Rework"

III.1.2.4. *Sector "Prueba del Sistema"*

Los errores que QA no detecta mientras se diseña y codifica el software y los errores malos que resultan de un rework defectuoso permanecen sin detectar hasta la etapa de prueba, donde se asume que *todos* serán detectados y corregidos, ya que en este modelo no se contempla la operación de mantenimiento.

Este sector (ver Figura III.6.) modela dos conjuntos de procesos:

a). El crecimiento de los errores no detectados.

b). La prueba del sistema que resulta en la detección y corrección de esos errores.

a). Los errores no detectados por QA y los arreglos malos que resultan de un rework defectuoso no permanecen inactivos hasta ser detectados y corregidos, sino que tienen una "existencia activa" produciendo más y más errores en el sistema. Por ejemplo, un error de diseño no detectado lleva a errores adicionales en el código.

Se sabe que detectar y corregir un error de diseño durante la fase de diseño consume un décimo del esfuerzo que sería necesario para detectarlo y corregirlo durante la fase de prueba porque también implicaría corregir inventario de especificaciones, código, manuales, etc. Mayor esfuerzo implica mayor costo.

Como no hay literatura que reporte la dinámica del proceso de "regeneración de error", se intenta describirla en el modelo de este sector.

Los errores no detectados serán "*Errores activos*" o "*Errores pasivos*". Todo error de diseño será error activo ya que se traducirá error de código. A medida que se avanza en el desarrollo aparecerán errores pasivos. Esta mezcla de errores que cambia a lo largo de la vida del proyecto se representa en el modelo con la variable "*Porcentaje de errores activos*".

Los "*Errores pasivos no detectados*" permanecen inactivos hasta que son detectados y corregidos en la fase de prueba. Sin embargo los "*Errores activos no detectados*" seguirán generando errores y por eso en el modelo se representa un ciclo positivo en el cual un aumento en el nivel "*Errores activos no detectados*" lleva a un incremento en la "*Tasa de regeneración de error activo*" la cual lleva a un mayor incremento en el nivel y así sucesivamente.

La "*Tasa de regeneración de error activo*" es una función de la "*Tasa de desarrollo de software*" dado que los errores sólo pueden ser generados cuando las tareas se

desarrollan. También la tasa de regeneración está en función de "*Densidad de error activo*" la cual es simplemente el número de errores activos dividido por las tareas desarrolladas hasta el momento. Más bien la tasa de regeneración en una función alisada de "*Densidad de error activo*", porque cuando se cometen errores en una parte del sistema, ellos en general, no afectarían a otras partes que están siendo desarrolladas en paralelo. Esto porque los errores tienden a propagarse a tareas subsiguientes que se construyen unas sobre otras, tales como las tareas de código que se construyen sobre diseños incorrectos. Así hay un retraso antes de que un error reproduzca errores adicionales. En el modelo este retraso se estableció a 3 meses. Por ejemplo, si un error en las primeras fases se reproduce en varias regeneraciones a un total de 9 errores, luego en la fase de prueba en vez de tratar con un error se tratará con 10 es decir, se ha producido una escalación de 10 veces en el costo.

La escalación en el número de errores activos se da de dos maneras:

- El error activo se alimenta a si mismo (bucle positivo). El error generado más temprano y no detectado, es el que más reproducciones producirá y por lo tanto terminará siendo el más costoso.
- Un "*Multiplicador de regeneración de error activo debido a la densidad de error*" representa el número promedio de nuevos errores que un simple error activo reproduce en una generación. Esto es una medida de la fertilidad del error. El valor del multiplicador siempre será mayor que 1 ya que un error no detectado generará siempre más de un error en una simple generación. El multiplicador aumenta cuando aumenta la densidad de errores activos. Así como aumenta la densidad de error, la distribución de errores también y se vuelven menos localizables y más caros para detectar y corregir. (Escalación en el costo).

Ya se indicó que "*Errores activos no detectados*" podrán continuar generando errores mientras nuevas tareas sigan desarrollándose. Pero no es así para todos los errores. Algunos errores no seguirán reproduciendo hasta el final de la fase de desarrollo, sólo lo hará por una o dos generaciones. Cuando un error cesa de reproducirse se vuelve un "*Error pasivo no detectado*". La tasa a la cual ocurre esto se denomina "*Tasa de retiro de error activo*".

Esta tasa es regulada a través de "*Fracción de retiro*" que es la fracción de errores activos que se vuelven pasivos por unidad de tiempo. Esta fracción es una función de la

fase de desarrollo. Debido a que cualquier error de diseño se traduce en errores de código, la "*Fracción de retiro*" permanece en 0 durante la fase de diseño dado que ningún error de diseño activo se retirará ni se volverá pasivo, en vez de ello, cada error de diseño producirá por lo menos una generación de errores de código. Cuando el proyecto progresa hacia las últimas fases de desarrollo la propagación de error rápidamente decrece y la "*Fracción de retiro*" consecuentemente aumenta y alcanza el valor 1 al final del desarrollo.

b). Cuando el proyecto se aproxima a las últimas fases de desarrollo, se inician las actividades de prueba del sistema. El objetivo de estas actividades es verificar que todos los elementos del sistema se integran apropiadamente y que se alcanzan todas las funciones y performance.

El cambio de asignación de mano de obra desde el desarrollo a la prueba se presenta en el modelo de "Asignación de Mano de Obra" por medio de la variable "*Fracción de esfuerzo para prueba del sistema*". El valor inicial de esta variable es 0 ya que el esfuerzo es asignado a la fase de desarrollo (ningún esfuerzo se asigna inicialmente a la prueba). Cuando se percibe que el desarrollo se completó, el valor se vuelve 1 ya que el 100% del esfuerzo disponible para el desarrollo o prueba es asignado a la función de prueba. Este cambio no es abrupto ya que hay solapamiento entre las fases de desarrollo y prueba.

La tasa a la cual se prueban las tareas desarrolladas esta determinada por el cociente entre la "*Mano de obra diaria para prueba*" y la "*Mano de obra de prueba necesaria por tarea*".

La "*Mano de obra de prueba normal necesaria por tarea*" tiene una componente fija (es independiente del número de errores e involucra la sobrecarga de actividades tales como planes de prueba, herramientas de prueba de instalación y diseño de casos de prueba) y una componente variable (es una función del número de errores en una tarea y representa el esfuerzo de prueba que sería gastado en la detección y corrección de errores).

La componente fija mencionada, "*Sobrecarga de prueba normal*" está definida en el modelo en términos de días/hombre / KDSI. Boëhm estima que este esfuerzo sobrecargado está sobre 2 días/hombre / KDSI.

Además de la sobrecarga en la que se incurre en probar una tarea, es necesario esfuerzo para detectar y corregir cualquier error restante. El esfuerzo es el producto de la "*Densidad de error*" y la "*Mano de obra de prueba normal necesaria por error*". El valor del primero se obtiene dividiendo la suma de los errores activos y pasivos restantes por el

número de tareas que todavía faltan probar. Esto representa el número promedio de errores por tarea. El valor de la "*Mano de obra de prueba normal necesaria por error*" se fijó en 0,15 días/hombre / error. Para el día normal de trabajo de 8 hs., el valor es 1,2 días/hombre / error.

El esfuerzo de prueba real necesario por tarea es una función no sólo de la sobrecarga de prueba y la densidad de error, sino también de cuan eficiente son los empleados. Por lo tanto hay que considerar si hay pérdidas por comunicación y motivación. Si hay tales pérdidas, la mano de obra actual necesaria será mayor que la mano de obra normal necesaria.

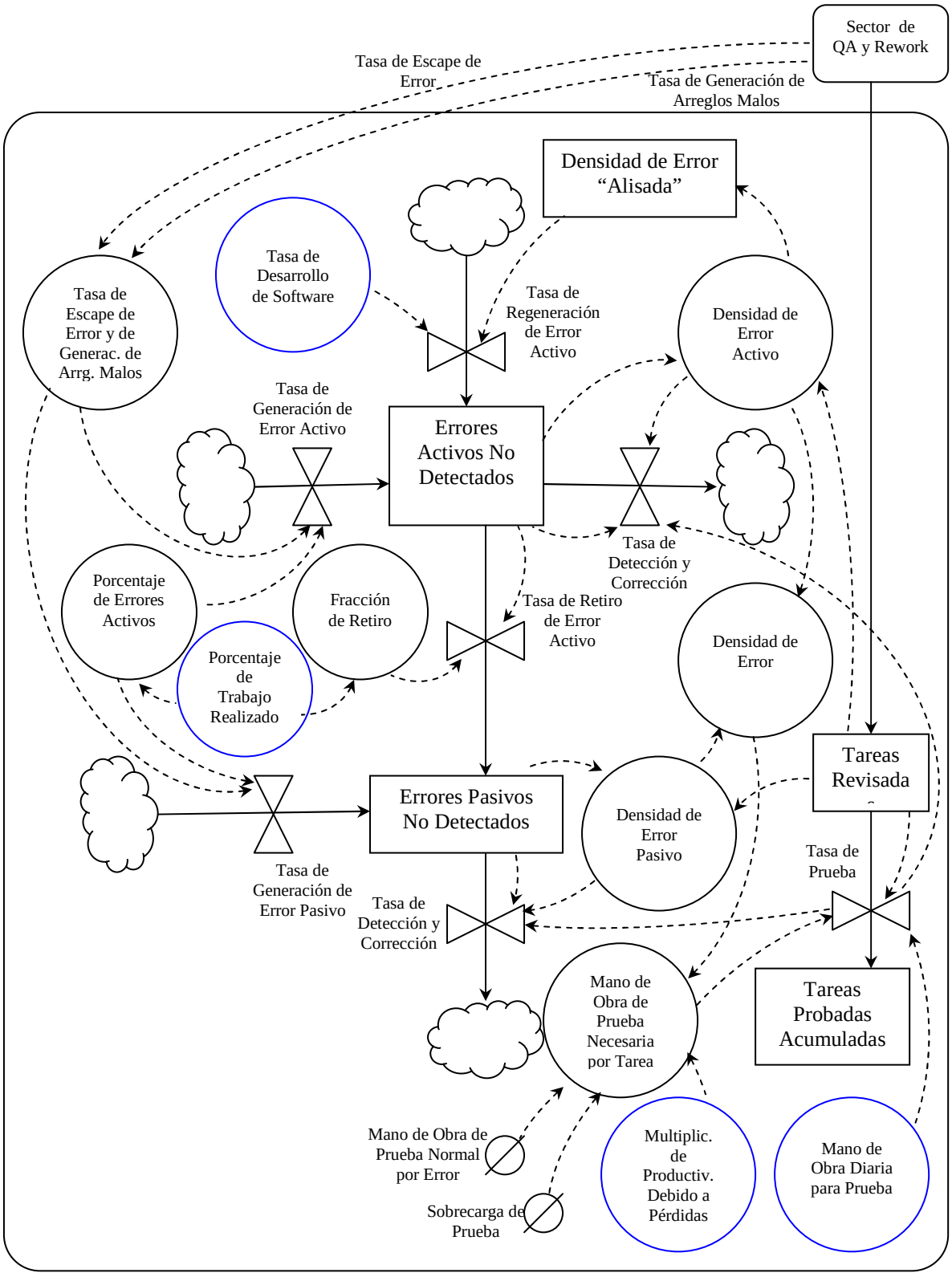


Figura III.6. Sector "Prueba de Sistema"

III.1.3. *Funciones de Gestión del Desarrollo de Software*

III.1.3.1. *Subsistema "Control"*

Cualquier función de control tiene tres elementos:

- *Medición*: detección de lo que está sucediendo en la actividad que está siendo controlada.
- *Evaluación*: comparación de la información de lo que actualmente está sucediendo con algún estándar o expectativa de lo que debería estar ocurriendo.
- *Comunicación*: informe de lo que ha sido medido y evaluado, para que la conducta pueda alterarse si la necesidad de hacerlo así lo indica.

Estos tres elementos están incluidos en el modelo ilustrado en las Figuras III.7. y Figura III.8.

El progreso en un proyecto software se mide por el número de recursos consumidos, tareas completadas o ambos. El "*Esfuerzo total percibido todavía necesario*" para completar el proyecto se determina desde estas medidas. El esfuerzo total incluye aquel percibido todavía necesario para desarrollar tareas de QA, de corrección (rework) de errores detectados y para completar la prueba del sistema. El esfuerzo percibido todavía necesario se compara con el "*Esfuerzo restante*" real en el plan del proyecto.

Una vez que se realiza la evaluación sobre cualquier escasez o exceso de esfuerzo, el comportamiento del proyecto puede alterarse si la necesidad de hacerlo así lo indica. Por ejemplo, si el proyecto se percibe retrasado, los miembros del proyecto pueden ser motivados para trabajar horas extras o se extiende el plazo del proyecto o ambas cosas.

A cualquier punto la cantidad de trabajo que será percibido como restante será en general una combinación de:

- Trabajo necesario para desarrollar y detectar errores (QA) de nuevas tareas.
- Trabajo necesario para corregir (rework) cualquier error detectado.
- Trabajo necesario para conducir las actividades de prueba.

Así el "*Esfuerzo total percibido todavía necesario*" para completar el proyecto es la suma de "*Esfuerzo percibido todavía necesario para nuevas tareas*", "*Esfuerzo percibido todavía necesario para rework errores detectados*" y "*Esfuerzo percibido todavía necesario para prueba*".

Debido a que el software es un producto intangible durante la mayoría del proceso de desarrollo, el progreso se mide, especialmente en las primeras fases, por la tasa de gastos de recursos más que por alguna cuenta de logros.

En este caso el informe del estado termina siendo un eco del plan original. En otras palabras "*Esfuerzo percibido todavía necesario para nuevas tareas*" es igual a "*Esfuerzo percibido restante para nuevas tareas*".

Pero cuando el proyecto avanza, y el trabajo se vuelve más evidente, las discrepancias entre el porcentaje de tareas logradas (restantes) y el porcentaje de recursos gastados (restantes) se vuelve cada vez más evidente.

Cuando el proyecto avanza hacia el final, los miembros del proyecto se vuelven cada vez más capaces de percibir cuan productiva ha sido realmente la mano de obra. Como un resultado, el valor de "*Esfuerzo percibido todavía necesario para nuevas tareas*" deja de ser una función "*Esfuerzo percibido restante para nuevas tareas*" y se vuelve función de lo que los miembros del proyecto perciben como cantidad de trabajo restante.

Estos modos diferentes de medir el progreso se modelan formulando "*Esfuerzo percibido todavía necesario para nuevas tareas*" como el cociente entre "*Tareas percibidas restantes*" y "*Productividad de desarrollo asumida*" ($C = D / E$), donde: "*Productividad de desarrollo asumida*" es un promedio pesado/ponderado de "*Productividad de desarrollo percibida*" y de "*Productividad de desarrollo proyectada*", es decir:

"*Productividad de desarrollo asumida*" = PDProyectada * W + PDPercibida * (1-W)

donde el factor de peso/ponderación W se mueve de 1 (al principio del proyecto) a 0 (al final de la fase de desarrollo).

- <u>*En las fases iniciales*</u> el progreso tiende a ser medido por la tasa a la cual los recursos son gastados. Como resultado el informe del estado termina siendo nada más que un eco del plan original y bajo tales condiciones "*Esfuerzo percibido todavía necesario para nuevas tareas*" es igual a "*Esfuerzo percibido restante para nuevas tareas*" ($F = C$). Sustituyendo C se tiene $F = D / E$ lo cual lleva a $E = D / F$.

  Así, esto sugiere que cuando los miembros miden e informan el progreso por la tasa de gastos de recursos, ellos están <u>*asumiendo*</u> que su productividad (E) es

igual a "*Tareas percibidas restantes*" (*D*) dividido por "*Esfuerzo percibido restante para nuevas tareas*" (*F*). Esto es interesante porque tal valor de productividad asumida es solamente una función de proyecciones futuras (tareas y esfuerzo restante) como opuesto a ser una reflejo de logros (tareas completadas y recursos gastados). Esta *notación implícita* de productividad es representada en el modelo por la variable "*Productividad de desarrollo proyectada*", definida, como sugiere la ecuación anterior, igual a "*Tareas percibidas restantes*" (*D*) dividido por "*Esfuerzo percibido restante para nuevas tareas*" (*F*).

Así, en las fases iniciales de desarrollo, la ecuación $C = D / E$ se vuelve $C = D / G$ donde $G = D / F$, la cual se logra estableciendo el valor del factor de peso *W* igual a 1 y sustituyendo en la ecuación $C = D / E$.

- Cuando el proyecto avanza <u>*hacia sus fases finales*</u> los logros se vuelven más visibles y los miembros del proyecto se vuelven más capaces de percibir cuan productivos han sido. Como resultado de esto, lo que los miembros asumen como su productividad, el valor de "*Productividad de desarrollo asumida*", deja de ser función de proyecciones futuras (tareas y esfuerzo restantes) y se vuelve función de logros percibidos. Esta *notación explícita* de productividad es representada en el modelo por la variable "*Productividad de desarrollo percibida*". Según entrevistas la productividad global del equipo, al final de la fase de desarrollo, se determina por la división entre "*Tareas desarrolladas acumuladas*" (*I*) y "*Esfuerzo de desarrollo acumulado*" (*J*). Así al final de la fase de desarrollo la ecuación $C = D / E$ se reduce a $C = D / PDPercibida$ (PDPercibida es productividad percibida) donde $PDPercibida = I / J$ la cual se alcanza estableciendo el factor de peso *W* en 0 y sustituyendo en la ecuación $C = D / E$.

Lo que los miembros asumen como su productividad cambia a medida que avanza el desarrollo, pero el cambio no es abrupto. La gradualidad del cambio de lo que asumen como productividad está modelada por el factor de peso *W*. La tasa a la cual *W* se mueve de 1 a 0 es el producto de dos factores: las tasa de gastos de recursos y la tasa de desarrollo de tareas. En el modelo *W* es el producto de dos multiplicadores: "*Multiplicador del peso de productividad debido a gasto de recursos*" y "*Multiplicador del peso de productividad debido al desarrollo*".

Ambos multiplicadores se mueven del valor 1, al principio del proyecto, al valor 0 cuando todos los recursos de desarrollo estimados son gastados o todas las tareas son desarrolladas.

El "*Esfuerzo percibido todavía necesario para rework errores detectados*" es el producto de "*Errores detectados*" y "*Mano de obra de rework percibida necesaria por error*". Éste último es una función alisada de "*Mano de obra de rework real necesaria por error*".

El "*Esfuerzo percibido todavía necesario para prueba*" se determina por el cociente entre "*Tareas restantes para probar*" y la "*Productividad de prueba percibida*". "*Tareas restantes para probar*" es la diferencia entre "*Tareas probadas acumuladas*" y "*Tamaño percibido del trabajo en tareas*".

A lo largo de la mayoría de la fase de desarrollo y antes del comienzo de la fase de prueba, el valor de "*Productividad de prueba percibida*" es igual a "*Productividad de prueba planeada*" que es el valor de productividad de prueba que está implícita en el plan del proyecto.

Sin embargo cuando el proyecto avanza, las percepciones de la gente sobre su productividad, se vuelve una función de cuan productiva es la actividad de prueba *real* en oposición a cuan productiva ha sido *planeada*. La "*Productividad de prueba real*" se determina por el cociente entre "*Tareas probadas acumuladas*" y "*Esfuerzo de prueba acumulado*". "*Productividad de prueba percibida*" se formula como una función alisada ya que los cambios raramente se hacen inmediatamente, hay una tendencia a retrasar la acción (ante una caída en la productividad, por ejemplo) hasta que el cambio es insistente. El retraso de alisado se fijó en 50 días de trabajo.

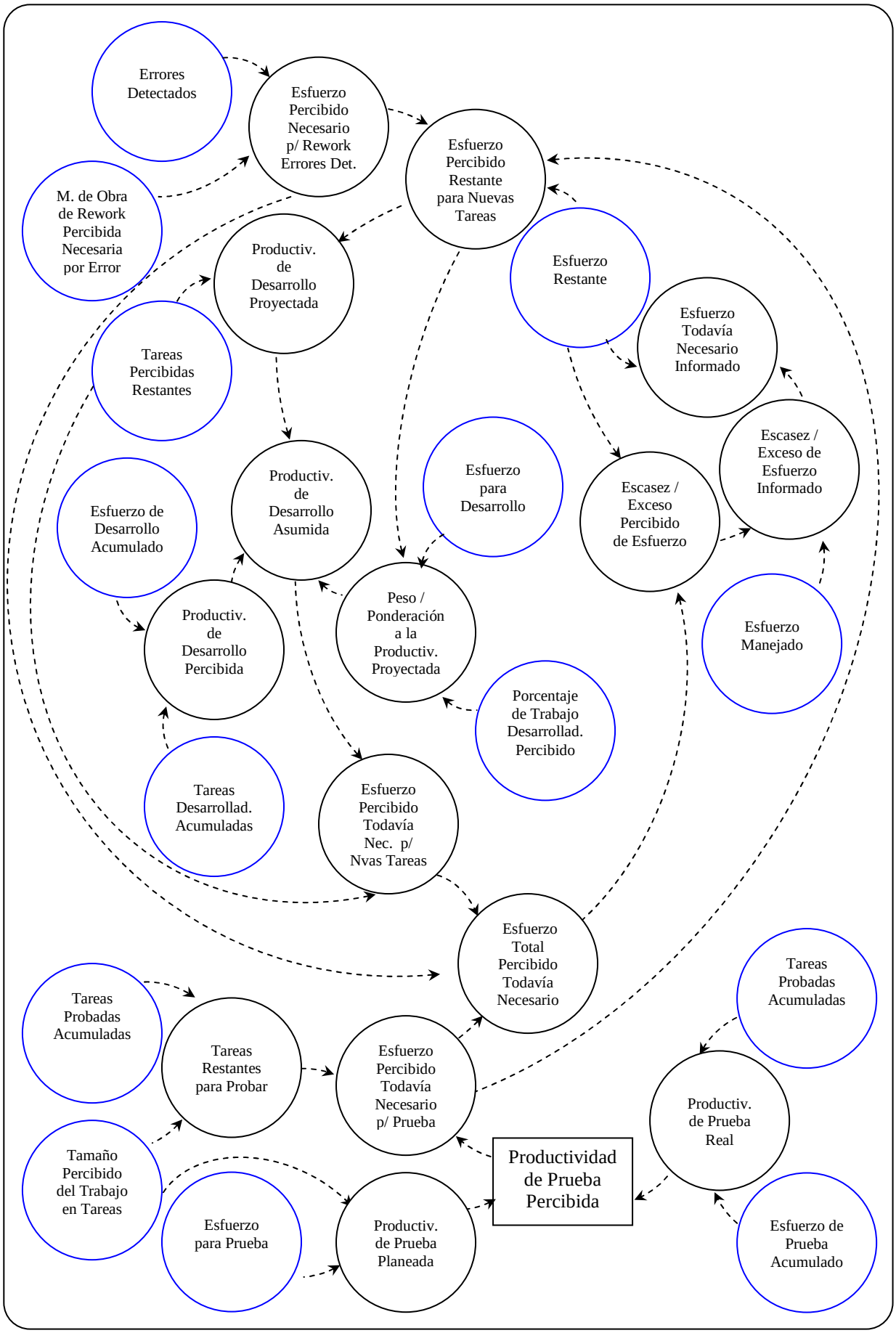


Figura III.7. Subsistema “Control”

Una vez que se determinan "*Esfuerzo percibido todavía necesario para nuevos tareas*", "*Esfuerzo percibido todavía necesario para rework errores detectados*" y "*Esfuerzo percibido todavía necesario para prueba*", se suman para determinar el "*Esfuerzo total percibido todavía necesario*" para completar el proyecto. Este total es comparado luego con el "*Esfuerzo restante*" real en el plan del proyecto.

Después de la evaluación se determina si hay escasez o exceso y el comportamiento del proyecto puede alterarse. Ya se indicó anteriormente, cuando se hizo referencia a productividad de desarrollo, cuáles son los mecanismos para determinar cuánto del exceso o escasez de esfuerzo se absorbe por los miembros del proyecto,

Cualquier escasez o exceso no absorbido será informado y lleva a ajustes en el alcance del proyecto. Tales ajustes son luego traducidos en ajustes al cronograma, ajustes en el nivel de mano de obra o ambos.

Lo informado como no absorbido se suma a "*Esfuerzo restante*" y la suma constituye el valor de "*Esfuerzo todavía necesario informado*" que sumado a "*Esfuerzo gastado acumulado*" permite ajustar el "*Tamaño total de trabajo en esfuerzo*".

El proceso de ajuste se informa por la "*Tasa de ajuste del tamaño del trabajo en esfuerzo*". Esta tasa es a la cual "*Tamaño total de trabajo en esfuerzo*" se ajusta hacia arriba o hacia abajo de lo que se percibió como su nuevo valor. Esta tasa está computada como:

(Meta – Nivel) / Tiempo de ajuste donde

Meta = valor revisado de tamaño del trabajo en esfuerzo

= Esfuerzo todavía necesario informado + Esfuerzo gastado acumulado

Nivel = Tamaño total de trabajo en esfuerzo

Tiempo de ajuste = Retraso en el ajuste del tamaño del trabajo en esfuerzo

Se fija ese retraso porque raramente el ajuste se hace primero, ya que cuando se detecta una escasez o exceso, se reacciona primero absorbiendo tal escasez o exceso. Sólo cuando esto no es suficiente se hace el ajuste. En el modelo este retraso se fijó en 3 días de trabajo.

Un retraso en el programa del proyecto no es la única razón para ajustar hacia arriba el tamaño del trabajo en *esfuerzo*. También puede suceder cuando se hace un ajuste hacia arriba en el tamaño del trabajo en *tareas*.

Puede hacerse una subestimación del tamaño en tareas. Por eso en el modelo se define el parámetro "*Fracción de subestimación de tareas*" que se utiliza para actualizar el valor de "*Tamaño del trabajo en tareas percibido actualmente*" y "*Tareas de trabajo no descubiertas*".

Cuando el desarrollo avanza, se descubren tareas progresivamente. El número de tareas no descubiertas que se descubren por unidad de tiempo es regulado por "*Tasa de descubrimiento de tareas*" la cual es el producto del número de "*Tareas de trabajo no descubiertas*" y de "*Porcentaje de tareas no descubiertas que se descubren por día*". Esta tasa de descubrimiento aumenta cuando se avanza en el desarrollo.

Las tareas descubiertas se incorporan en el proyecto (incorporadas en el WBS, Gant o PERT) y esto lleva tiempo, por eso la "*Tasa de incorporación de tareas descubiertas en el proyecto*" se modelo como un retraso de tercer nivel y el "*Retraso promedio de incorporación de tareas descubiertas*" se fijó en 10 días de trabajo.

El descubrimiento de tareas se traduce en adiciones de asignación de esfuerzo del proyecto.

Las tareas descubiertas no necesariamente llevan a ajustes sobre la estimación de esfuerzo del proyecto. Primero el número de tareas descubiertas se clasifican según tamaño dividiéndolo por "*Productividad de desarrollo asumida*" y se obtiene el "*Tamaño percibido de las tareas descubiertas en esfuerzo*". Pero este valor *absoluto* del tamaño de la tarea descubierta no es suficiente por si mismo para definir si se realiza el ajuste, también se requiere conocer su valor *relativo* de tamaño: "*Tamaño relativo de las tareas descubiertas*" que es igual al cociente entre "*Tamaño percibido en esfuerzo de las tareas descubiertas*" y "*Esfuerzo percibido restante para nuevas tareas*". Este valor es luego comparado con un valor umbral: "*Máximo tamaño relativo de adiciones toleradas sin agregar esfuerzo al proyecto*". Si el tamaño relativo es menor que el umbral, las tareas recién descubiertas son totalmente absorbidas sin ajustar el esfuerzo estimado del proyecto. Caso contrario parte de todas las tareas adicionales se traducen en esfuerzo adicional en el plan del proyecto. En el modelo se fijó a 1% el tamaño máximo relativo.

La decisión de realizar el ajuste (incorporar ya sea todas o una parte de las tareas descubiertas en la estimación del esfuerzo del proyecto) involucra dos estimaciones, una para el esfuerzo de desarrollo y QA de las nuevas tareas y otra para la prueba del sistema. La primera se hace dividiendo el número de tareas descubiertas que serán incorporadas por

"*Productividad de desarrollo asumida*" y la última, dividiendo por "*Productividad de prueba percibida*". Cualquier ajuste de la estimación del esfuerzo total del proyecto llevará a ajustes ya sea en la fecha de compleción o en el nivel de la mano de obra o ambos.

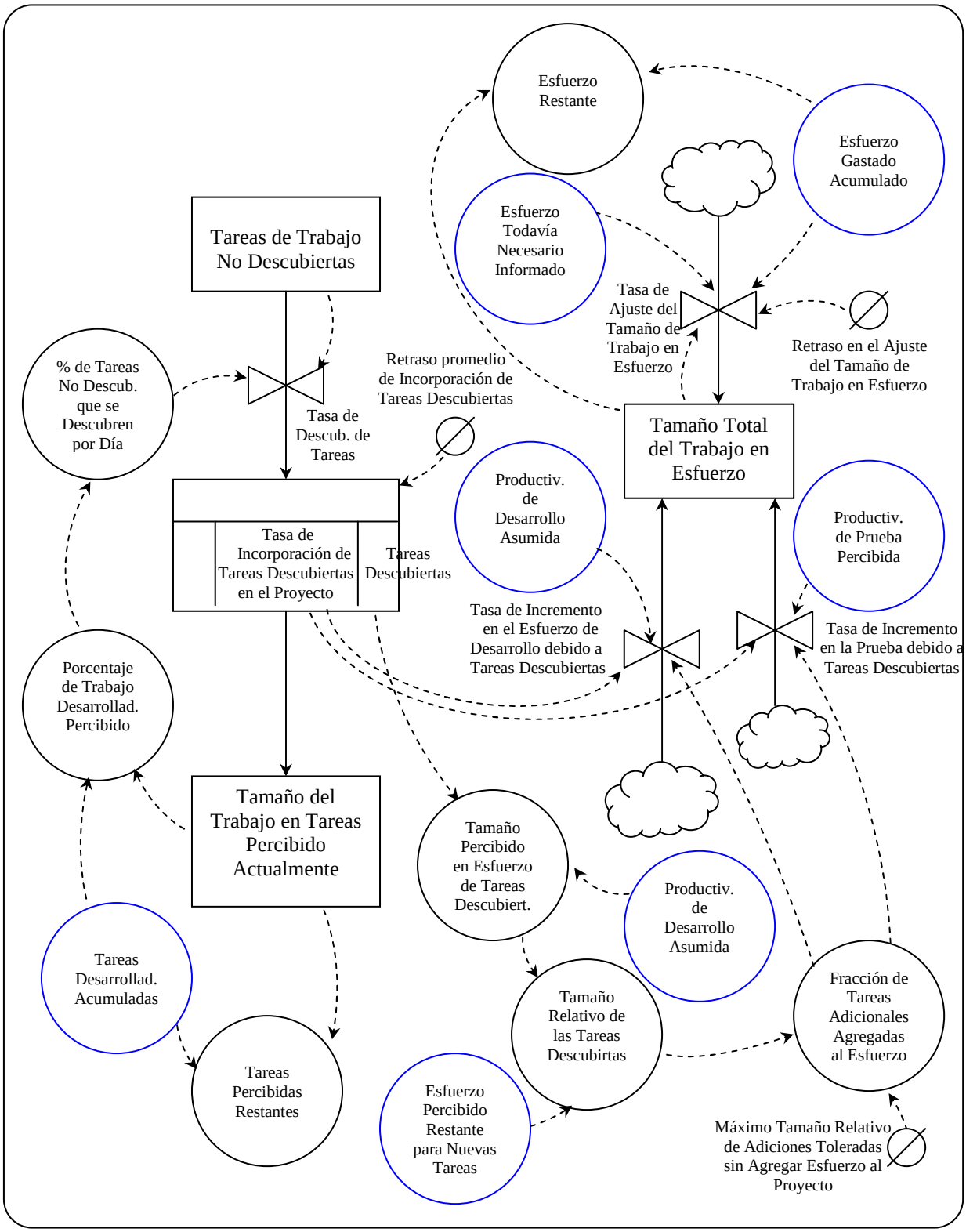


Figura III.8. Ajuste del Tamaño del Trabajo

III.1.3.2. *Subsistema "Planificación"*

En este subsistema se hacen estimaciones iniciales para comenzar el proyecto y luego esas estimaciones se revisan cuando es necesario a lo largo de la vida del proyecto. El modelo de este subsistema se muestra en la Figura III.9.

La "*Fecha de compleción programada*" no representa una fecha real sino el número de días de trabajo desde el comienzo del proyecto. Así, restándole el valor actual de "*Tiempo*" (que representa el número de días de trabajo ya corridos en la simulación) se puede determinar el "*Tiempo restante*". Dividiendo el valor de "*Esfuerzo restante*" por "*Tiempo restante*" se puede determinar el "*Nivel de mano de obra indicada*" a cualquier punto del proyecto. Este nivel representa el número de empleados full-time que se cree necesario y suficiente para completar el proyecto a tiempo de acuerdo a la actual "*Fecha de compleción programada*".

Pero si los empleados no son asignados full-time, deben realizarse ajustes, dividiendo el valor de "*Nivel de mano de obra indicada*" por el valor de "*Mano de obra diaria promedio por empleado*".

Si el "*Nivel de mano de obra indicada*" es menor que "*Total de mano de obra*" los empleados en exceso se transfieren fuera del proyecto. Si "*Nivel de de mano de obra indicada*" es mayor se deben contratar más empleados. (Como de indicó en el Subsistema Gestión de Recursos Humanos).

El "*Mano de obra necesaria*" es un promedio pesado/ponderado del actual "*Total de mano de obra*" y de "*Nivel de mano de obra indicada*". Esto responde al nivel de mano de obra estable y al número de empleados que serían requeridos para completar el proyecto a tiempo:

$$\textit{Mano de obra necesaria} = \textit{Nivel de mano de obra indicada} * \textit{WCWF} + \textit{Total de mano de obra} * (1 - \textit{WCWF})$$

Esta formulación sólo se aplica cuando el valor de "*Nivel de mano de obra indicada*" es mayor que "*Total de mano de obra*" indicando necesidad de contratar más empleados. Cuando ocurre lo contrario, "*Mano de obra necesaria*" se establecería al valor más bajo y cualquier exceso de empleados serían transferidos fuera del proyecto.

El factor de peso/ponderación *WCWF* (intención de cambiar el nivel de mano de obra) asume valores entre 0 y 1. Cuando *WCWF* es igual a 1 el peso considera sólo el "*Nivel de mano de obra indicada*" y la gestión ajusta su nivel de mano de obra al número percibido requerido para finalizar en tiempo. Cuando *WCWF* se mueve hacia 0, más y más peso se da a la estabilidad de la mano de obra. Cuando *WCWF* es igual a 0 el número pesado de empleados deseado depende totalmente de la estabilidad de la mano de obra.

WCWF consiste de dos componentes. El primero, *WCWF1*, representa la presión que desarrolla para la estabilidad de la mano de obra cuando el proyecto procede hacia sus etapas finales.

Existe una "*Máxima fecha de compleción tolerable*". Con tal que "*Fecha de compleción programada*" esté por debajo de esa fecha máxima, decisiones de ajuste de tiempo, más contrataciones o combinación de éstas continuarán basándose en balance de tiempo y estabilidad de mano de obra como lo indicado en *WCWF1*. Sin embargo cuando "*Fecha de compleción programada*" comienza a aproximarse a "*Máxima fecha de compleción tolerable*" hay presión sobre las consideraciones de la estabilidad de la mano de obra. Esto es, la gestión se vuelve más deseosa de pagar cualquier precio por evitar superar ese máximo y en esos casos la gestión desea contratar más gente.

El desarrollo de tal presión de tiempo es representado por *WCWF*

WCWF = Máximo(WCWF1, WCWF2)

Cuando "*Fecha de compleción programada*" está por debajo de "*Máxima fecha de compleción tolerable*" el valor de *WCWF2* es igual a 0, es decir, no tiene efecto sobre la determinación de *WCWF* y consecuentemente no hay presión sobre decisiones de contratar. Cuando "*Fecha de compleción programada*" se aproxima al máximo, el valor de *WCWF2* comienza a aumentar gradualmente. Debido a que esto se da hacia el final del proyecto, probablemente el valor de *WCWF1* es 0 y decrece. Así *WCWF2* excede a *WCWF1* y *WCWF* se vuelve totalmente dominado por el tiempo con la meta de no sobrepasar el máximo.

WCWF es una expresión de una política de la gestión del proyecto. Puede representar diferentes estrategias de cómo balancear ajustes de mano de obra y tiempo a lo largo del proyecto para minimizar desbordes de tiempos y costos.

"*Mano de obra necesaria*" determina si se contratan o se transfieren empleados. El número ajustado es "*Mano de obra buscada*" que casi siempre es idéntico a "*Mano de obra necesaria*". Cuando no es así, usualmente en las fases iniciales del proyecto, la tasa de construcción de mano de obra tiende a su más alto nivel. "*Mano de obra buscada*", en efecto, define el tope del número de empleados a ser contratados. Esto es, el nivel "buscado" podría establecerse al valor "necesario" con tal que sea menor o igual que el tope. Por otro lado, "*Mano de obra buscada*" se establece al valor de éste último.

Dividiendo el "*Esfuerzo restante*" por "*Mano de obra buscada*" (en términos de empleados full-time) se determina "*Tiempo percibido todavía requerido/restante*" el cual representa el tiempo restante, en días de trabajo, percibido requerido para completar el proyecto.

Tomando el "*Tiempo percibido todavía requerido/restante*" como una función de "*Mano de obra buscada*" en lugar de "*Total de mano de obra*" se asume que se realizan ajustes en el tiempo con pleno conocimiento de las decisiones de contratación que se hacen en el proyecto.

El "*Tiempo percibido todavía requerido/restante*" se suma al valor de "*Tiempo*" para determinar la "*Fecha de compleción indicada*". "*Fecha de compleción indicada*" se usa para ajustar "*Fecha de compleción programada*".

La "*Tasa de ajuste del plazo*" se calcula de forma similar que para ajustar el "*tamaño de trabajo*":

(Meta – Nivel) / Tiempo de ajuste donde

Meta = fecha de compleción indicada

Nivel = fecha de compleción programada

Tiempo de ajuste = tiempo de ajuste del plazo (que en el modelo se fijó en 5 días de trabajo).

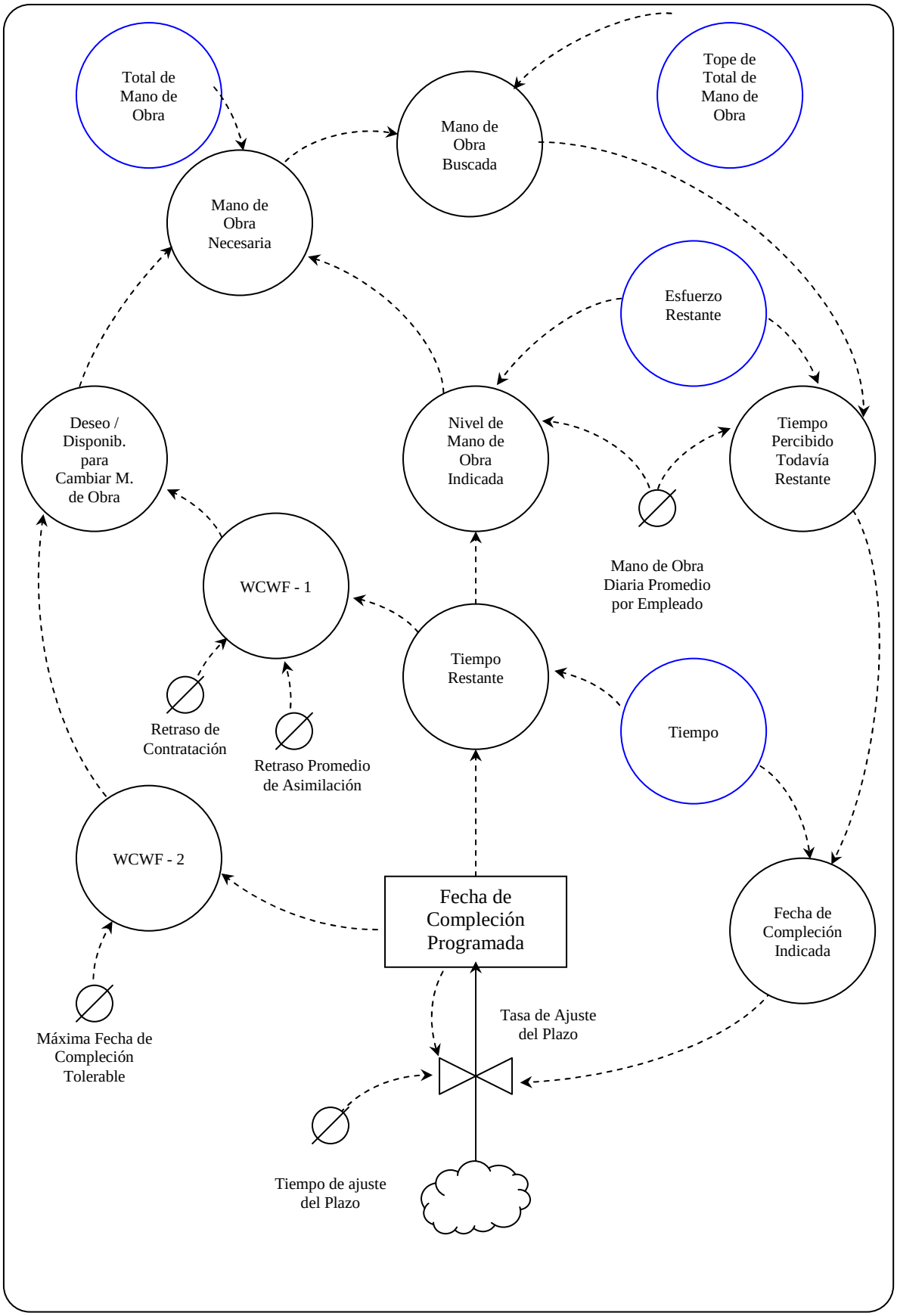


Figura III.9. Subsistema “Planificación”

CAPITULO IV

. . . . . .

EXTENSIÓN DEL MODELO DE ABDEL-HAMID Y MADNICK

Presentación

Un proceso de desarrollo de software global requiere de un sólido proceso de *gestión del proyecto software.* Particularmente, en el marco del presente trabajo, se sostiene que una adecuada *gestión de requisitos* puede formar la base para la estimación, planificación y control del progreso del proyecto para soportar este tipo de desarrollo y propiciar el éxito de dicho proyecto.

Así, reconociendo las bondades de los modelos de simulación para guiar el proceso de gestión de proyectos software, en el presente Capítulo se formaliza el *Modelo Dinámico Extendido para la Gestión de Proyectos Software*, el cual toma como base el modelo de Abdel-Hamid y Madnick, descrito en Capítulo III. El *Modelo* propuesto adiciona la etapa de Análisis, omitida en el modelo base, para lograr un modelo integrado que pueda aplicarse globalmente a cada una de las etapas del proceso de desarrollo de software para dar soporte a la toma de decisiones, en el marco de la gestión de proyectos.

A continuación se describe el *Modelo* propuesto para la etapa de análisis y la interfase que lo integra con el modelo base.

IV.1. *Modelo Propuesto de la Etapa de Análisis*

En esta sección se desarrolla, de acuerdo a la metodología de la Dinámica de Sistemas [20], el modelo propuesto de la etapa de análisis del proceso de desarrollo de software, el cual integra las siguientes actividades generales: *elicitación, análisis, especificación y validación de requisitos*. Al mismo tiempo contempla, explícita o implícitamente, una diversidad de *factores* que intervienen en el dinamismo de tales actividades, como el efecto de la presión de tiempo, la asignación de la mano de obra al análisis, la productividad de la mano de obra asignada, el cambio de requisitos, el surgimiento de requisitos nuevos, la generación de errores durante el análisis, etc.

Las Figuras IV.1, IV.2, IV.3, IV.14 y IV.15 muestran diferentes niveles de formalización del modelo propuesto.

IV.1.1. *Descripción general del sistema, identificación de elementos y relaciones fundamentales*

El modelo correspondiente a la etapa de análisis se basa en el flujo de tareas, desde la obtención de los *requisitos del cliente* hasta la compleción de las *especificaciones*. Es decir, en la etapa de análisis hay una transformación desde tareas incompletas, cuando los requisitos se elicitan a través de actividades conjuntas con el cliente, hasta tareas completas, que se logran con la producción de las especificaciones. Esta idea general que caracteriza la etapa de análisis, vista como una caja negra, se sintetiza en la Figura IV.1.

Figura IV.1. Descripción General de la Etapa de Análisis

Durante el transcurso de esta etapa, una fracción de las especificaciones puede no ser aceptable, por contener *errores* o porque se les debe incorporar *cambios* que surgen con posterioridad a su definición inicial, y requieren que se las consideren en un proceso iterativo de corrección (ciclo de rework / ciclo de gestión).

Otra fracción de especificaciones puede no ser aceptable, por considerarse *obsoletas* o *duplicadas*, y se rechaza por no ser adecuada a las necesidades de los clientes.

Por otra parte, el modelo contempla también la posibilidad del *surgimiento de nuevos requisitos* después de iniciada la etapa y que deben incorporarse al proceso de análisis.

La *producción de las especificaciones* de requisitos está en función de la *mano de obra asignada* a la etapa de análisis y de cuan productivos son esos recursos asignados. La mano de obra asignada a esta etapa es un porcentaje del *esfuerzo total* asignado al proyecto

y la *productividad* se mide de forma análoga al modelo base (Capítulo III - Apartado III.1.2.2). Estas dos últimas variables constituyen dos de las vinculaciones con el modelo de Abdel-Hamid y Madnick.

Una vez que las especificaciones han sido validadas, pasan a formar parte del flujo de entrada para la fase de desarrollo. Esto establece otra relación con el modelo de Abdel-Hamid y Madnick.

Las vinculaciones con el modelo base se detallan más adelante al describir el diagrama de Forrester y en el apartado IV.2.

La Figura IV.2 permite visualizar los flujos básicos de la etapa de análisis, según la descripción precedente.

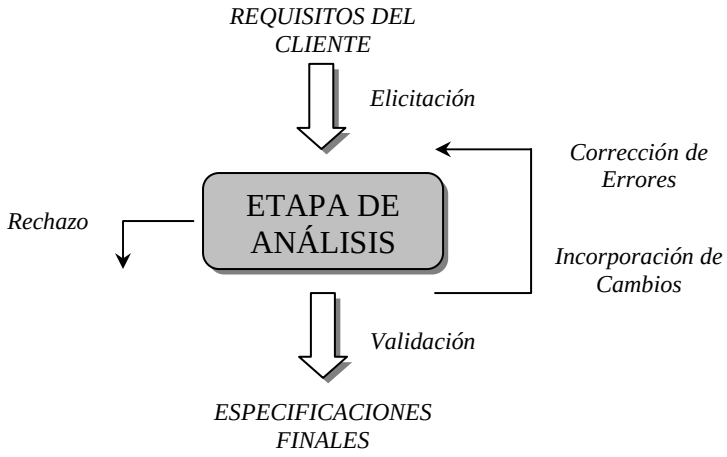


Figura IV.2. Descripción de Flujos Básicos de la Etapa de Análisis

Una descripción más detallada se formaliza luego al presentar el Diagrama Causal y el Diagrama de Forrester.

IV.1.2. *Construcción del Diagrama Causal*

En este paso del desarrollo del modelo, se seleccionan los factores que ayudan a visualizar cómo el proceso de gestión de requisitos influye en el proceso global de gestión del proyecto software. Dichos factores fueron seleccionados desde la literatura consultada.

El diagrama causal de la etapa de análisis se construyó para mostrar los factores en estudio que participan en esta fase y que, se supone, influyen en el proceso global de gestión del proyecto software, en relación especial con el costo y tiempo insumidos para alcanzar el éxito del proyecto, medido en última instancia en concepto de calidad.

La interpretación general que permitió derivar este diagrama es la siguiente:

A partir de estimaciones del tamaño del producto, que definen la *funcionalidad* del proyecto, se obtienen *estimaciones iniciales de tiempo y costo* con lo cual se procede a la contratación de la *mano de obra* necesaria. A medida que el proyecto avanza se obtiene información sobre el *progreso* del mismo. Comparando los datos de la situación actual con los datos planificados, puede llevar a alguna modificación en las estimaciones de costo y tiempo. Esta modificación puede involucrar más contrataciones de recursos humanos, que por lo general se da cuando se detecta que el proyecto se encuentra retrasado.

La *presión de tiempo* es un factor importante que con frecuencia ocurre en proyectos retrasados y tiende a neutralizarse con la contratación de nueva mano de obra y/o trabajo extra. La presión juega un rol motivacional significativo sobre la *productividad*. En un principio, un aumento en la presión contrae el tiempo de ocio de los miembros del equipo, por lo que aumenta la productividad pero también, luego de un tiempo, como consecuencia del estrés y agotamiento incrementa la tasa de generación de error, puesto que bajo presión se trabaja más rápido pero no necesariamente mejor. A su vez, al aumentar la cantidad de *errores,* aumenta también el esfuerzo dedicado al *rework* y a la *prueba*, en este caso disminuyendo la productividad y por lo tanto se extiende el *tiempo de desarrollo*. Extender el tiempo estimado, a su vez aumenta la presión.

Particularmente, al extender el *tiempo de la etapa de requisitos* más tiempo se da para que surjan *nuevos requisitos* ya que es más probable que el entorno y/o los usuarios cambien como consecuencia de una maduración del conocimiento adicional logrado del mismo desarrollo o por presiones del entorno o de la organización. Esto conlleva cambios que aumentan la cantidad de rework que debe realizarse, dando lugar a arreglos defectuosos lo que implica incremento de *requisitos inadecuados* que a su vez requerirá mayor *cambio de requisitos*. El trabajo extra que significa realizar los cambios de requisitos, lleva a mayores niveles de presión.

Así como la presión tiene efectos negativos y positivos sobre la productividad, ocurre lo mismo con la contratación de *mano de obra*. Al aumentar la mano de obra crece la *complejidad de la comunicación*, porque aumentan las líneas de intercambio entre los miembros del equipo. Así ni los *empleados recién contratados* ni los *experimentados* alcanzan su máxima productividad ya que los novatos primero deben pasar por un período de formación llevado a cabo por los experimentados, quienes deberán suspender sus propias tareas. Entonces el efecto positivo de contratar más mano de obra sólo ocurrirá luego de finalizar el período de formación.

Resumiendo, se puede destacar cómo, según aumente o disminuya la *productividad*, aumenta o disminuye el *tiempo de desarrollo* y esto a su vez hace que aumente o disminuya la *presión*. Así los ciclos que involucran estos factores clave se reinician *reforzándose* o *equilibrándose*.

En el diagrama causal, que se muestra en la Figura IV.3, se representa la importancia de cada factor en forma conjunta con la relación establecida con el resto de los factores.

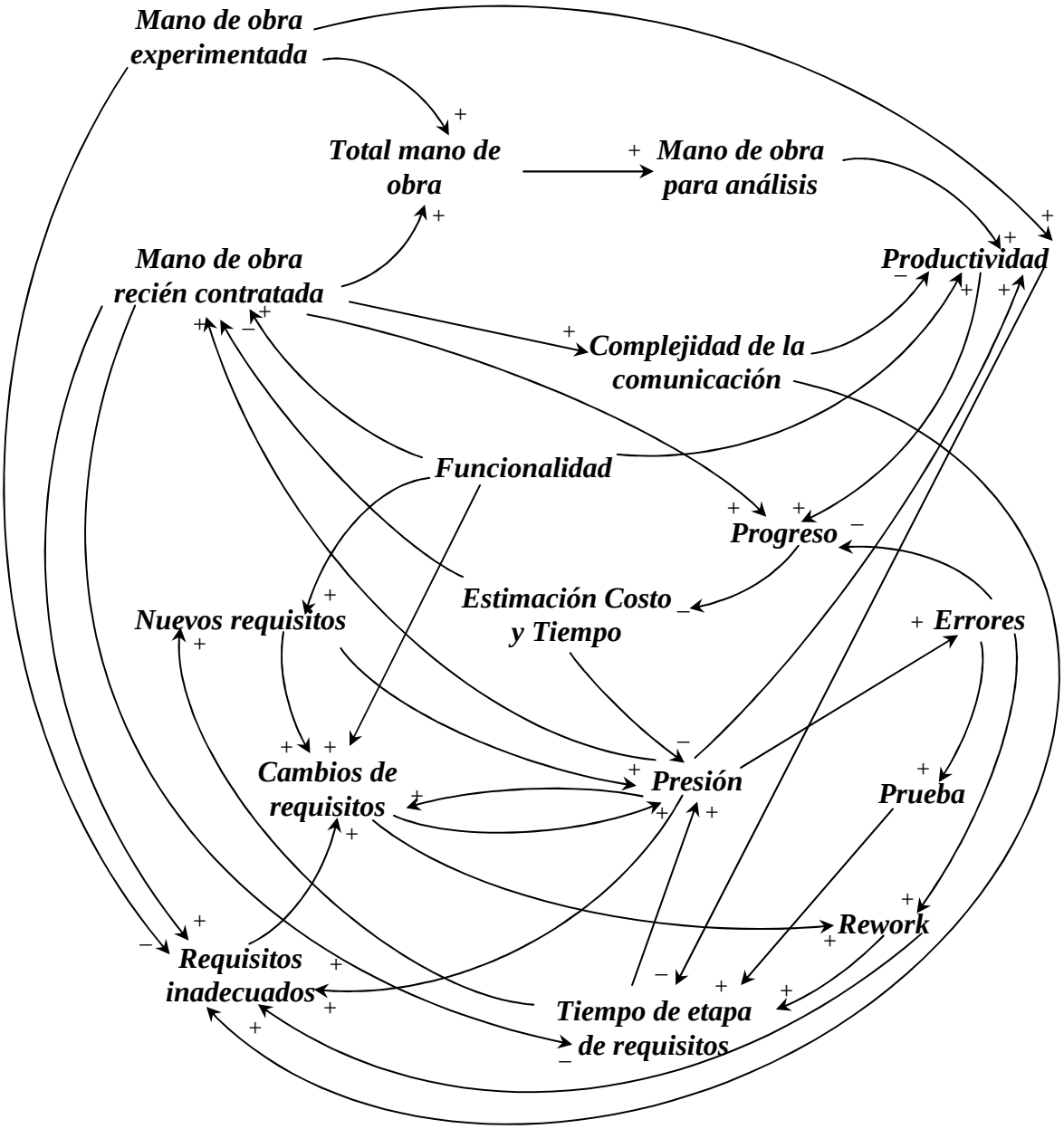


Figura IV.3. Diagrama Causal del Modelo Propuesto para la Etapa de Análisis

A continuación se explican detalladamente los factores incluidos en el diagrama de causal.

- *Mano de Obra recién contratada:* número de empleados recientemente contratados para trabajar en el proyecto.
- *Mano de obra experimentada:* número de empleados con conocimiento sobre el dominio actual para el trabajo del proyecto.
- *Total de mano de obra:* número total de empleados afectados al trabajo del proyecto. Es la suma de "mano de obra recién contratada" y "mano de obra experimentada".
- *Mano de obra para análisis:* número de empleados afectados al trabajo de la etapa de análisis del proyecto.
- *Productividad:* es una medida del trabajo de especificación de requisitos producido por empleado por unidad de tiempo.
- *Complejidad de la comunicación:* medida de las líneas de comunicación establecidos entre los implicados en el proyecto.
- *Nuevos requisitos:* es una medida de la solicitud de nuevos requisitos.
- *Cambios de requisitos:* es una medida de los cambios realizados en las especificaciones de los requisitos.
- *Requisitos inadecuados:* es una medida de la calidad de las especificaciones de los requisitos.
- *Presión:* es una medida del efecto del proyecto retrasado en el plazo de tiempo.
- *Funcionalidad:* es una medida del tamaño y complejidad del software estimados para ser desarrollada.
- *Tiempo de etapa de requisitos:* es el tiempo requerido para producir las especificaciones de los requisitos elicitados.
- *Errores:* es la cantidad de errores generados en la etapa de análisis.
- *Prueba:* es el esfuerzo gastado o insumido para corregir errores.
- *Rework:* es el esfuerzo gastado o insumido para trabajar los cambios de requisitos.
- *Estimación de costo y tiempo:* estimaciones reguladas por el progreso del proyecto y que inciden en la adquisición de recursos humanos y en el tipo de presión de plazo.
- *Progreso:* es una medida del avance/estado del proyecto

Para profundizar en el entendimiento de las relaciones descriptas en forma general previo a la construcción del diagrama causal, se procede a la descripción detallada de algunas relaciones.

Al mismo tiempo que se construye el diagrama causal se pueden determinar *diagramas de ciclos o bucles causales* a partir de los cuales es posible obtener un entendimiento básico de las relaciones de causa-efecto entre los factores considerados.

Examinar esas relaciones en forma aislada, generalmente, puede resultar fácil de comprender, sin embargo cuando éstas se combinan en una larga cadena de causa-efecto en el diagrama causal, se vuelven complejas. Los diagramas de ciclos o bucles causales incrementan la comprensión de esas relaciones complejas.

Si bien el diagrama causal precedente corresponde al *modelo de análisis*, integra factores que constituyen la estructura del modelo base, con lo cual es posible observar la interfase entre ambos.

Es oportuno describir previamente esta *estructura básica* aislando sus cuatro bucles de realimentación [1]:

a). El ciclo que se presenta a continuación en la Figura IV.4 regula el nivel de recursos humanos en función del progreso percibido en el proyecto:

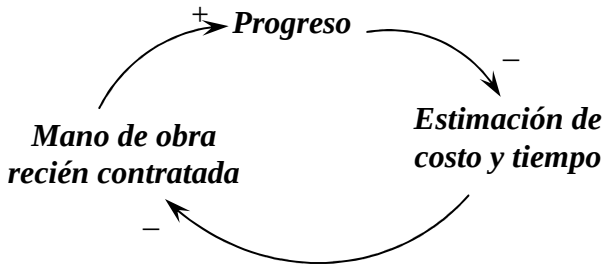


Figura IV.4. Relación de Causalidad: Nivel de Mano de Obra–Progreso Proyecto

Como en los modelos estáticos, a partir de estimaciones del tamaño del producto, se obtienen estimaciones iniciales de tiempo y costo. De éstas últimas se procede a la contratación de la mano de obra necesaria. A medida que el proyecto avanza se obtiene información sobre el progreso del mismo. La comparación los datos de la situación actual con los datos planificados, puede llevar a alguna modificación en las estimaciones de costo y tiempo. Tal modificación puede involucrar más contrataciones de recursos humanos, que por lo general se da cuando se detecta que el proyecto se encuentra retrasado.

b). El siguiente bucle de la Figura IV.5 muestra el efecto de la presión y de los errores sobre el progreso del proyecto:

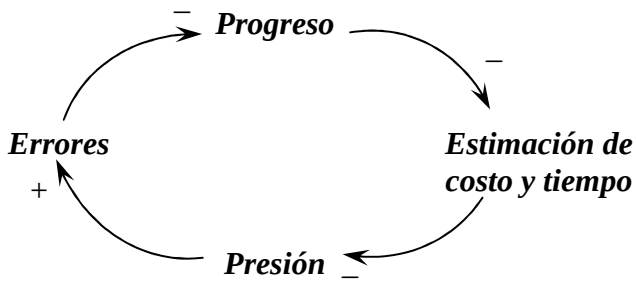


Figura IV.5. Relación de Causalidad: Presión–Progreso Proyecto

Ya se indicó, en el Capítulo precedente, que la presión se obtiene de la comparación entre la fecha de entrega planificada y la fecha de entrega estimada. Ésta última, depende de la percepción real del progreso del proyecto. Si la fecha de entrega estimada es mayor que la planificada (presión positiva) el proyecto se encuentra retrasado. En éstos casos la presión se tiende a neutralizar mediante nuevas contrataciones de mano de obra y/o trabajo extra.

Sin embargo, el trabajo extra mantenido por un período prolongado produce agotamiento, con lo cual la generación de error es mayor que en una situación de trabajo normal. Este aumento en la tasa de generación de errores incide sobre el progreso del proyecto ya que cuanto mayor sea el número de errores, mayor será el esfuerzo dedicado a las tareas de detección y corrección de error, lo que produce una desviación de esfuerzo desde las actividades de producción hacia las actividades de corrección y por lo tanto las actividades puramente productivas disminuyen. Esta disminución del progreso lleva a nuevos ajustes en las estimaciones de costo y tiempo para tratar de neutralizar la presión.

c). El efecto de la contratación de mano de obra nueva sobre la productividad del equipo, se ilustra en la Figura IV.6.:

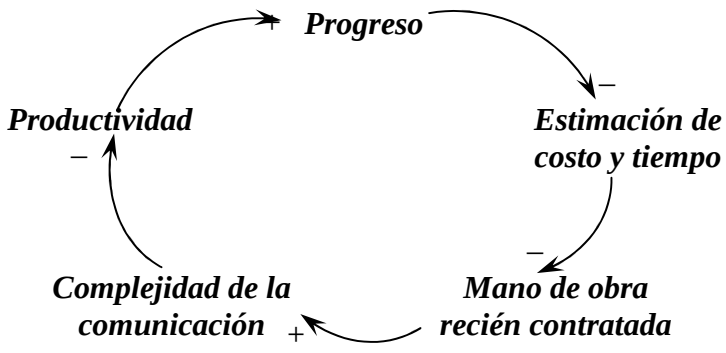


Figura IV.6. Relación de Causalidad: Nivel de Mano de Obra-Productividad

Es común considerar que un incremento en la mano de obra se traducirá en un incremento en la productividad media del equipo y que, por lo tanto, también aumentará el progreso del proyecto. Pero, como lo señalan los autores del modelo base, la contratación tiene efectos negativos sobre las variables del proyecto, ya que la relación entre el número de miembros del equipo y la productividad no es lineal. Esto es porque, incorporar nuevos empleados produce pérdidas de productividad ya que, por un lado aumenta las líneas de comunicación y, por otro lado, el personal nuevo normalmente no tiene la misma productividad que los empleados experimentados.

Por eso los empleados recién contratados pasan por un período de formación para conocer las particularidades del proyecto y de la organización. Pero como las actividades de formación son llevadas a cabo por el personal experimentado, éstos no estarán dedicados completamente a sus tareas específicas, por lo que su productividad desciende. Luego a medida que el período de formación se completa, la productividad del equipo comienza a crecer motivada por el incremento de la productividad del personal nuevo y porque disminuyen las actividades de formación del personal experimentado.

d). En la Figura IV.7. se observa el efecto de la presión sobre la productividad:

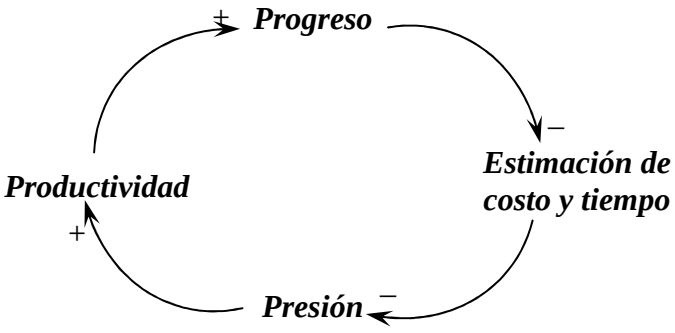


Figura IV.7. Relación de Causalidad: Presión – Productividad

Como se indicó anteriormente, cuando la presión es positiva (el proyecto está retrasado) aumenta la productividad trabajando horas extras o por la eliminación de tiempos muertos (tiempo de ocio). Pero esta situación lleva al agotamiento de los empleados y sus consecuencias ya señaladas.

Si la presión es negativa (proyecto adelantado) los empleados se relajan, por lo cual su productividad disminuye y esto lleva a consumir la sobreestimación del plazo. Por esto, si el valor de la presión negativa es muy alto, se revisan las estimaciones para adaptarlas al valor estimado.

Como se mostró para la estructura del modelo base, también se pueden aislar los siguientes diagramas de ciclos o bucles causales que involucran factores específicos del *modelo de análisis*:

a). El diagrama siguiente, Figura IV.8, ilustra cómo la presión de plazo afecta el tiempo gastado en la fase de requisitos:

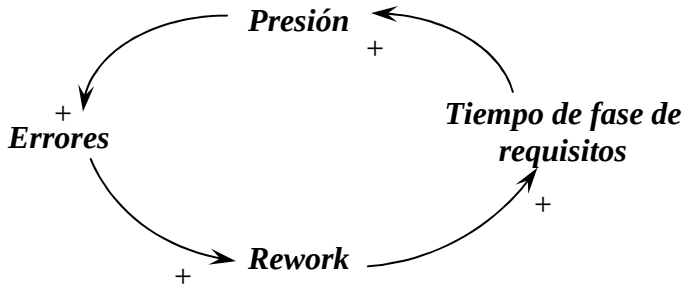


Figura IV.8. Relación de Causalidad: Presión – Tiempo de Fase de Requisitos

Un incremento en la presión, incrementa la generación de error debido al alto nivel de tensión, stress, agotamiento, que produce en los miembros del equipo. Una densidad de error elevada aumenta la cantidad necesaria de esfuerzo y tiempo de rework (y también de prueba) que se consumen de las actividades de producción y por lo tanto aumenta el tiempo de la fase de requisitos, el cual a su vez incrementa la presión.

b). Por otro lado se constituye el siguiente ciclo, como se ve en la Figura IV.9:

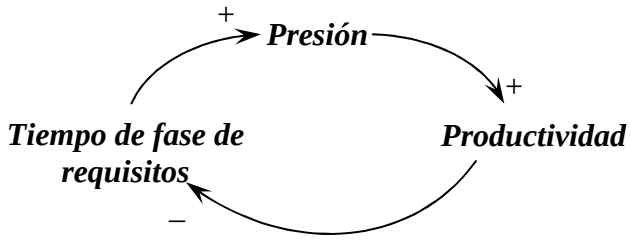


Figura IV.9. Relación de Causalidad: Productividad–Tiempo de Fase de Requisitos

Como se indicó anteriormente, bajo presión (positiva) incrementa la productividad debido a su efecto de motivación (trabajando horas extras o absorbiendo tiempos de ocio). Al aumentar la productividad disminuye el tiempo insumido en la fase de requisitos, y tal progreso a su vez, hace reducir la presión.

c). También se puede observar, en la Figura IV.10, el próximo diagrama de ciclo o bucle causal en el que participan los siguientes factores:

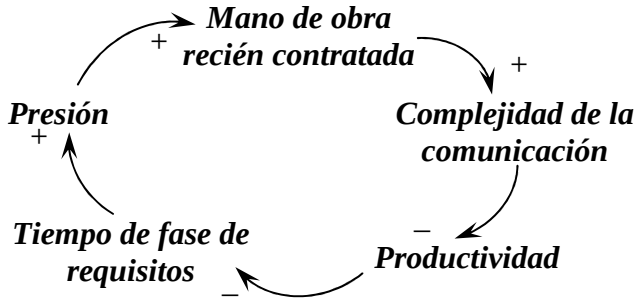


Figura IV.10. Relación de Causalidad: Nivel de Mano de Obra – Tiempo de Fase de Requisitos

A mayor presión, una política del jefe del proyecto puede ser contratar mano de obra nueva, la cual pasa por un período de orientación, a cargo de personal experimentado, tanto en dimensiones técnicas como sociales relacionadas al proyecto, período durante el cual los empleados son menos productivos. Al disminuir la productividad, aumenta el tiempo de fase de requisitos y esto hace que aumente la presión.

d). Otro bucle causal queda determinado por los siguientes factores, como se observa en la Figura IV.11.:

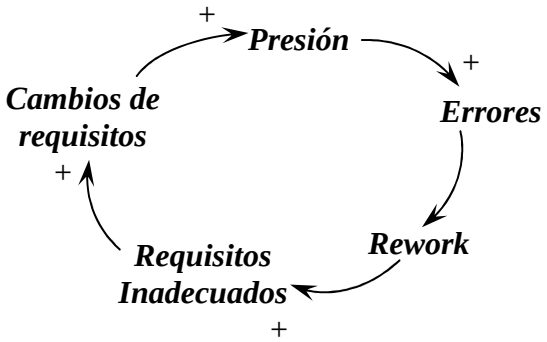


Figura IV.11. Relación de Causalidad: Cambios de Requisitos–Producción de Errores

Realizar cambios de requisitos lleva a mayores niveles de presión ya que significa trabajo extra. Al aumentar la presión, se trabaja más rápido pero no necesariamente mejor, por lo que aumenta la producción de errores que a su vez incrementa el esfuerzo dedicado a rework. Las correcciones realizadas llevan a arreglos defectuosos, lo que implica aumento de requisitos inadecuados que a su vez requerirá mayor cambio de requisitos.

e). También se pudo aislar el ciclo que se muestra debajo, en la Figura IV.12:

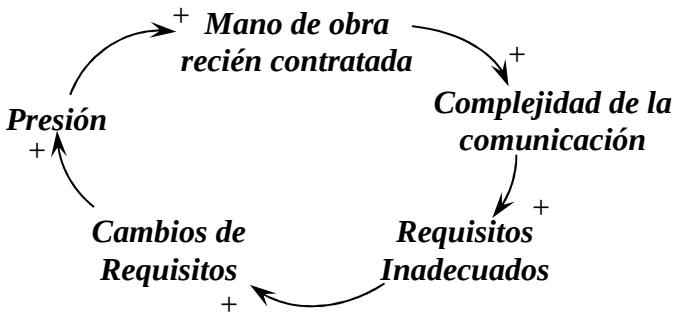


Figura IV.12. Relación de Causalidad: Nivel de Mano de Obra – Cambios de Requisitos

Cuando la dirección del proyecto decide contratar nueva mano de obra, aumentan las líneas de comunicación, y mientras dure el período de formación y adaptación, las actividades de producción, tanto de los empleados experimentados (que llevan a cabo la formación) como los empleados nuevos (que todavía no completan el conocimiento de las singularidades del proyecto), pueden llevar a un aumento en el desarrollo de requisitos inadecuados, lo que a su vez lleva a mayores cambios de requisitos.

El trabajo extra que significa realizar los cambios de requisitos, lleva a mayores niveles de presión, que puede implicar la decisión de contratar más mano de obra, con lo cual se completa el ciclo.

f). Por último, en la Figura IV.13, se observa el siguiente bucle causal:

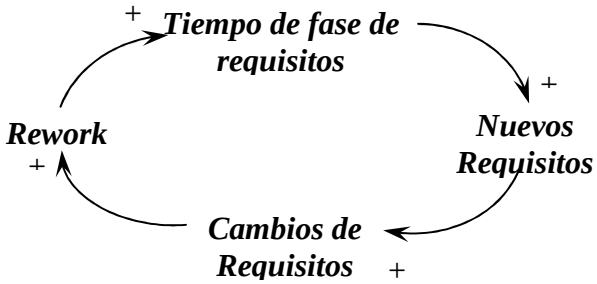


Figura IV.13. Relación de Causalidad: Surgimientos de Nuevos Requisitos– Tiempo de Fase de Requisitos

A medida que avanza el proyecto, pueden surgir nuevos requisitos de usuario como consecuencia de una maduración del conocimiento adicional logrado del mismo desarrollo o por presiones del entorno o de la organización. Esto puede llevar a cambios en los requisitos originales que sí fueron contemplados desde los primeros pasos de la educción, lo que a su vez aumenta la cantidad de rework que debe realizarse sobre el avance en el

diseño y código logrado sin contemplar los nuevos requisitos y los cambios posteriores a su inclusión. Este trabajo de corrección (rework) aumenta el tiempo de la fase de requisitos y cuanto más se extiende, más tiempo hay para que surjan nuevos requisitos ya que es más probable que el entorno y/o los usuarios cambien.

IV.1.3. *Construcción del Diagrama de Forrester*

En esta instancia de construcción del modelo, y de forma análoga a la construcción del diagrama causal, pero en un nivel de formalización mayor, se describen las conexiones entre los factores de interés pero ya clasificados, de acuerdo con la convención de la Dinámica de Sistemas, como variables de flujo, de nivel o auxiliares. También se agregan algunos parámetros que se requieren para controlar los flujos y para obtener un adecuado funcionamiento del modelo que ayuda a visualizar cómo el proceso de gestión de requisitos influye en el proceso global de gestión del proyecto software.

También se describe la definición de las variables y parámetros incluidos en el modelo. Cabe destacar que para evitar obtener un modelo demasiado complejo, no fueron incluidos todos los factores del diagrama causal. Algunos factores se incluyen indirectamente en el modelo en forma de variables auxiliares o parámetros. Tal es el caso del factor de complejidad de la comunicación, el cual es incluido en la "Productividad de Desarrollo". También la cantidad de errores de análisis esta considerada con la inclusión de la variable "Medida de Exactitud de la Especificación".

Para una comprensión previa del diagrama de Forrester se proporciona la siguiente vista general como caja negra, con un esquema de bloques, donde el recuadro mayor corresponde al *modelo propuesto de la etapa de análisis* y los recuadros menores, diferenciados en color, se corresponden con subsistemas, sectores y/o subsectores del *modelo base* que contienen las variables con las que se relaciona e integra el modelo propuesto, ya sea a través de flujos de material o de información.

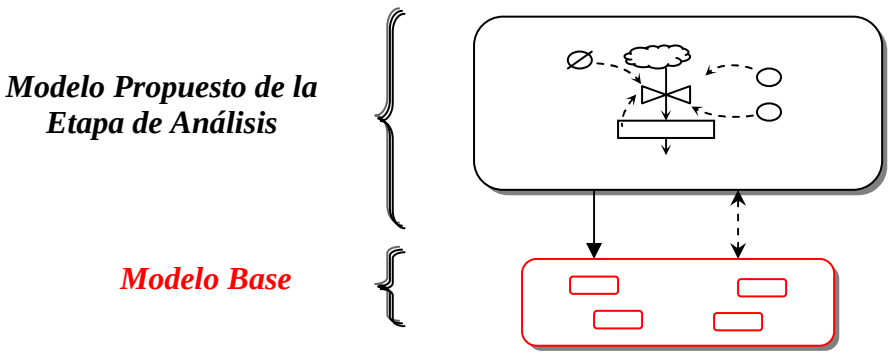


Figura IV.14. Esquema de Boques del Diagrama de Forrester

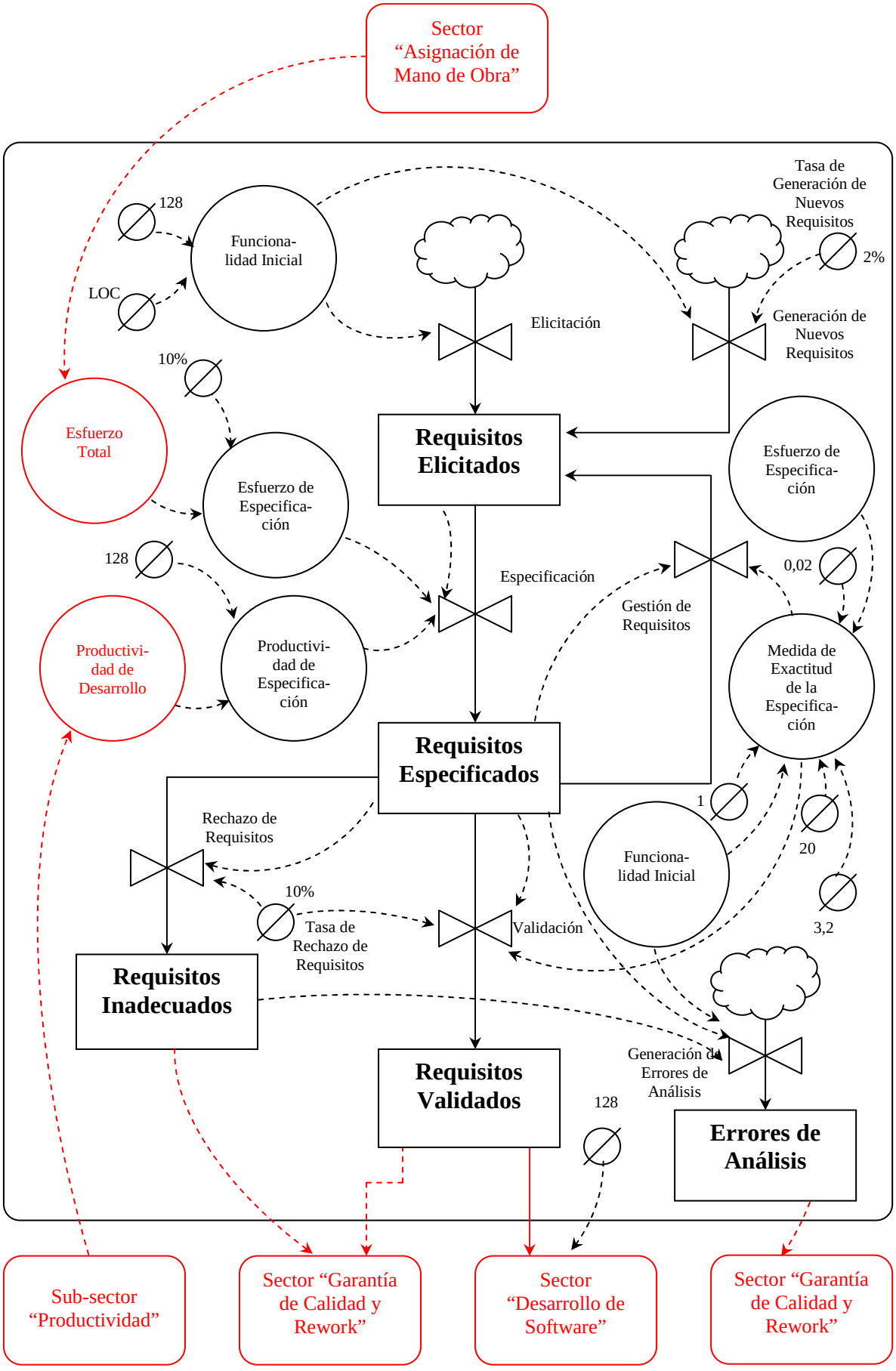


Figura IV.15. Diagrama de Forrester del Modelo Propuesto de la Etapa de Análisis

IV.1.3.1. *Descripción del Diagrama de Forrester*

En el marco de la Ingeniería de Software la meta es traducir la necesidad de un cliente, en un producto operativo. Para conseguir esta meta se debe crear una arquitectura y una infraestructura. La *arquitectura* comprende cuatro componentes de sistemas distintos: software, hardware, datos (base de datos) y personas. La *infraestructura* incluye la tecnología requerida para unir los componentes y la información que se emplea para dar soporte a los componentes.

Una visión general de tal arquitectura e infraestructura se consigue a través del *análisis del sistema.*

Esta actividad, según [26], se corresponde con la primera de tres fases genéricas que se asocian al trabajo de Ingeniería de Software y que la denomina *fase de definición*, en la que es fundamental identificar los requisitos claves del sistema y del software.

Así, la Ingeniería de Requisitos, como disciplina de la Ingeniería de Software, se centra en la identificación del propósito, la dirección y el alcance del sistema. Esto consiste en un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requerimientos del sistema. Es un proceso que abarca la elicitación, modelado, especificación y validación de requisitos.

En el *Modelo* propuesto, construido sobre en el conocimiento ganado desde la exploración bibliográfica y tomando como base el modelo de Abdel-Hamid y Madnick, la etapa de análisis basada en el flujo de tareas incluye las siguientes actividades:

a).- Elicitación de requisitos

b).- Análisis de requisitos

c).- Especificación de requisitos

d).- Validación de requisitos

e).- Gestión de requisitos

Estas actividades precedentes que caracterizan la etapa de análisis, suponen un proceso gradual en el cual una necesidad inicial manifestada por el cliente va

avanzando por una serie de transformaciones hasta alcanzar una representación precisa como requisitos de un producto.

Esta transformación sugiere diversidad de estados y formatos que abarca esa necesidad planteada. Inicialmente, a partir de la necesidad expresada verbalmente o por escrito por parte del cliente, se derivan los requisitos de la aplicación que se registran, generalmente, en lenguaje natural por parte del personal del equipo de desarrollo que está a cargo de la *elicitación de requisitos*. Se genera un documento introductorio con los requisitos del usuario. Este documento es fuente para posteriormente proceder al *análisis de requisitos* propiamente dicho. Éste implica una actividad de modelado con técnicas y metodologías específicas que transforma los requisitos originales, en lenguaje natural, en una expresión propia de la técnica y/o metodología utilizada.

Seguidamente, a partir de la representación alcanzada de la actividad de análisis, se construye lo que se conoce como *especificación de requisitos*, un documento formal que contiene una descripción detallada y precisa de los requisitos del sistema. Este documento sirve de base para un contrato entre el cliente y el desarrollador del software y constituye la base para las posteriores etapas de diseño, codificación y prueba del sistema. El producto derivado de esta actividad es una especificación de requisitos del sistema, luego de completada la *validación.*

Niveles

Esta serie de transformaciones que va "atravesando la necesidad del cliente" se define en el modelo propuesto, a través del paso por tres ***niveles*** principales:

- "*Requisitos Elicitados"*: recoge los requisitos del cliente.
- "*Requisitos Especificados"*: es el nivel donde los requisitos especificados se acumulan antes de ser clasificados/identificados como inadecuados o incorrectos o validados.
- "*Requisitos Validados"*: acumula las especificaciones una vez que se especifican en forma completa, coherente, precisa, sin ambigüedades y no conflictiva. Luego de esto, el nivel se vacía para transferir la funcionalidad a la fase de desarrollo.

Un cuarto nivel, *"Requisitos Inadecuados"*, acumula la fracción de las especificaciones que se descubren obsoletas o duplicadas y que se descartan por no ser acordes a las necesidades de los clientes. La cantidad *"Requisitos Inadecuados"* puede utilizarse como una medida de la calidad de las especificaciones que afecta la cantidad de código defectuoso que se produce en las fases de diseño y codificación.

El nivel, *"Errores de Análisis"*, acumula los errores generados en la etapa de análisis. Se incluye a los fines de poder analizar cómo la dinámica de la detección y corrección de los errores de análisis influyen en los tiempos y esfuerzos insumidos en actividades de QA (Garantía de Calidad - Quality Assurance), rework y prueba y en consecuencia en la reducción de la escalación del costo (al reducir el número de errores -densidad de error- que pasa de una etapa a la otra, se reduce también la regeneración de errores activos).

Flujos

Existen flujos que regulan la acumulación o movimiento de los niveles mencionados anteriormente y que los hacen crecer o decrecer.

Según lo indica la metodología Dinámica de Sistemas, la forma característica de definir una variable de flujo es:

$F(KL) = N(K) * FN * M(K)$ donde:

F es la variable de flujo
N es el nivel
FN es el flujo normal
M es un multiplicador del flujo normal (se usa para alterar el comportamiento normal de la variable)

Es con esta fórmula como se define el flujo "*Rechazo de Requisitos*", la cual controla la cantidad de especificaciones inadecuadas (obsoletas o duplicadas) y que en el modelo se fijó en 10% de los "*Requisitos Especificados*", ya que, según estadísticas tomadas de [8], ese es el valor que se da en casos normales.

Pero, usualmente, se hace difícil definir algunas ecuaciones de variables de flujo sin realizar algún cálculo con la ayuda de variables auxiliares que son combinaciones de información de entrada, y permiten clarificar y simplificar dichas definiciones. Tal es el caso de las variables "*Especificación*", "*Validación*", "*Generación de Nuevos Requisitos*", "*Gestión de Requisitos*" y "*Generación de Errores de Análisis*". A continuación de describen detalladamente las singularidades de la definición de estos flujos:

- "*Elicitación*": alimenta el nivel "*Requisitos Elicitados*" con el tamaño del trabajo inicial a desarrollar en la etapa de análisis y se define a partir de la variable auxiliar "*Funcionalidad Inicial*". Ésta última, como ya se indicó al detallar los factores incluidos en el diagrama causal, es una medida del tamaño inicial del software estimado para ser producido.

  Si bien la medida de productividad más comúnmente utilizada es la de "líneas de código / hombre/día", es insignificante comparar la productividad para diferentes lenguajes de programación, dado que la funcionalidad varía entre lenguajes, aún para el mismo número de líneas de código [8].

  Por esto, en el modelo en vez de medir la productividad en líneas de código, se usa la medida de funcionalidad mejor conocida: *puntos de función* [8]. Hacer esta elección permite hacer uso de una de las denominadas "reglas de oro" de [8] la cual admite transformar la cantidad de líneas de código (LOC) en la unidad puntos de función (PF) según la siguiente ecuación:

$$PF \approx \frac{LOC}{128}$$

  Entonces, para resumir, la variable "*Funcionalidad Inicial*" permite adaptar (de líneas de código a puntos de función) la información de entrada del tamaño del producto a ser desarrollado, disponible desde el proyecto seleccionado como caso de estudio, para dar comienzo a la producción de la etapa de análisis.

Es oportuno señalar que a partir de esta conversión, puntos de función es la unidad de medida de las especificaciones de requisitos que se utilizara a lo largo de la etapa de análisis.

- "*Especificación*": controla la producción de las especificaciones y, al igual que la "*Tasa de Desarrollo*" en el modelo de Abdel-Hamid y Madnick, esta definida como función de la mano de obra asignada a la actividad (en este caso la actividad de especificación) y de la productividad de esa mano de obra asignada.

  La mano de obra asignada a la tarea de especificación, denominada "*Esfuerzo de Especificación*", se determina como una proporción del "*Esfuerzo Total*" (valor de variable tomado del modelo base) asignado al proyecto. En el modelo esa proporción se fijó en un 10% según promedio calculado a partir de referencias citadas en [19] que sugieren porcentajes de división de esfuerzo para las distintas fases del ciclo de vida.

  Por su parte, la "*Productividad de Especificación*" se asume definida idénticamente a la "*Productividad de Desarrollo*" del modelo base, como "LOC / hombres/día", pero adecuada a la unidad de medida de puntos de función con la ayuda de un parámetro para obtener la cantidad de PF producidos por hombres/día.

- "*Validación*": controla la cantidad de especificaciones que se consideran adecuadas (validadas) y que ya no requieren pasar por el ciclo de gestión de requisitos ya que se considera que están definidas en forma completa, coherente, precisa, sin ambigüedades y no conflictiva.

  Para definir este flujo se asume que, por unidad de tiempo, llega al nivel "*Requisitos Validados*" la fracción de los requisitos especificados luego de haber descartado los que, en esa misma unidad de tiempo, pasan a ser inadecuados y gestionados.

- "*Generación de Nuevos Requisitos*": controla la llegada de nuevos requisitos, y se estableció en un valor promedio de 2% por mes basado

en información tomada de [8] que indica esa tasa de crecimiento de nuevas características y que deben agregarse a las características originales. Por supuesto este crecimiento debe detenerse en algún instante, pues de lo contrario el desarrollo se haría indefinido. Esto implica, de alguna manera, "congelar" la llegada de nuevos requisitos. En este caso, efectuando el control de que reste 15 días para finalizar la etapa de requisitos [8]. También podría permitirse continuar la generación de nuevos requisitos durante cierta cantidad de días, después de finalizada la etapa de requisitos.

- "*Generación de Errores de Análisis*": regula la acumulación de los errores de análisis en el nivel "*Errores de Análisis*". El valor de este flujo depende de la proporción de requisitos defectuosos definida por cociente "*Requisitos Inadecuados*" / "*Funcionalidad Inicial*".

Un flujo especial: Gestión de Requisitos

Un flujo especial, porque conforma el "eje" en el que convergen variables claves que determinan un ciclo de cambios de requisitos y que se supone influye en el proceso global de gestión del proyecto software, en relación especial con el costo y tiempo insumidos en el desarrollo del proyecto.

La definición de un conjunto de requisitos coherente, completo, no ambiguo y no conflictivo, no es lo que generalmente caracteriza el comienzo de un proceso de desarrollo de software. Los requisitos son resultado de muchas contribuciones con distintas y a veces contrapuestas necesidades y objetivos. Aquí es necesario priorizar las necesidades de las comunidades implicadas.

Los cambios en los requisitos de los usuarios frecuentemente son causantes de los desbordes en costos y tiempos en muchos proyectos software y de lo cual generalmente se culpa a los usuarios.

La complejidad de la definición de requisitos se incrementa de acuerdo a la duración del proyecto [10]. La razón se debe a que cuanto más largo es el proyecto, más probable es que el entorno del software, los usuarios y los clientes cambien. Los cambios en los requisitos serán solicitados a lo largo del ciclo de desarrollo, y se debe evaluar su costo y planificar su impacto en el trabajo hecho y en el pendiente. Si antes de las fases siguientes al análisis no se han identificado todos

los requisitos, tendrá un impacto muy negativo sobre el costo y la planificación posterior.

En los primeros pasos de la educción, lo requisitos están o poco especificados o sobreespecificados. Estos requisitos deben ser refinados en sucesivas iteraciones. Se deben contemplar los aspectos de contexto (organización, entorno, proyecto) para evitar llegar a requisitos incompletos, no verificables, innecesarios y no utilizables.

Esta característica de *volatilidad* que se da en los requisitos es porque durante el desarrollo de un sistema las necesidades del usuario pueden madurar a causa del conocimiento adicional fruto del desarrollo o de presiones del entorno o de la organización que no son previstos (cambian la complejidad de las organizaciones, los objetivos, políticas, funciones de los usuarios, etc.). Si no se incorporan estos cambios los requisitos originales serán incompletos e inconsistentes con la nueva situación. Entonces, si las necesidades de los usuarios evolucionan, el proceso de educción, análisis, especificación y validación se debe realizar más de una vez y las especificaciones obtenidas se van refinando de acuerdo al nuevo conocimiento obtenido.

Bajo estas apreciaciones, en el modelo propuesto se define por medio del flujo "*Gestión de Requisitos*" un ciclo de cambios de las especificaciones que permite modelar el mecanismo a aplicar para llevar a cabo el proceso iterativo de modificaciones para resolver el problema de volatilidad.

Particularmente, este flujo representa la cantidad de especificaciones que necesitan corregirse (rework) ya sea por contener errores o porque se les debe incorporar cambios que surgen con posterioridad a su definición inicial, para adecuarlas a la nueva situación. El valor de este flujo se determinó a partir de lo que se [8] define como *medida de la exactitud de la especificación*, dada por la ecuación:

$$1 - \frac{Tamaño\ estimado\ del\ \ producto\ a\ desarrollar}{N°\ \ de\ días\ \ para\ la\ \ fase\ de\ análisis\ de\ requisitos} * 0{,}02$$

La fracción indica la cantidad de tiempo disponible para especificar cada punto de función, la cual puede adoptarse como una medida de la calidad.

La constante 0,02 se determinó a partir de un cálculo empírico.

El numerador de la fracción se corresponde con lo que, en el modelo propuesto, ya se definió como "*Funcionalidad Inicial*" (medida del tamaño inicial del software estimado para ser producido).

El denominador de la fracción precedente, N° de días para la etapa de análisis de requisitos, esta determinado por el cociente entre:

$$\frac{Esfuerzo\ asignado\ a\ la\ etapa\ de\ análisis\ de\ requisitos * 20}{N°\ de\ personal\ para\ la\ etapa\ de\ análisis\ de\ requisitos}$$

La constante 20 indica la cantidad de días de trabajo al mes, que no considera los fines de semana, sino sólo los días hábiles.

El esfuerzo asignado a la etapa de análisis de requisitos, que es la proporción del esfuerzo total del proyecto que se asigna a la tarea de análisis, se corresponde con lo que en el presente modelo ya se definió como la variable "*Esfuerzo de Especificación*".

El número de personal para la etapa de análisis de requisitos se estableció como un valor fijo de 3,2 (tomado del modelo base), ya que se asume que durante la etapa de análisis de requisitos el retraso en el proyecto todavía no se descubre y no se necesita personal extra.

IV.1.4. *Definición de magnitudes: Código de variables*

Esta etapa implica construir una tabla que contiene la descripción, el símbolo nemotécnico, el tipo y la unidad de medida (magnitud) para cada una de variables incluidas en el diagrama de Forrester (ver Tabla IV.1.).

Nº de Orden	Símbolo Nemotécnico	Descripción	Tipo de Variable	Magnitud
1	ReqElicitados	Requisitos elicitados	Endógena / Nivel	PF
2	ReqEspecificad	Requisitos especificados	Endógena / Nivel	PF
3	ReqValidados	Requisitos validados	Endógena / Nivel	PF

Tabla IV.1. Definición de variables del modelo propuesto

Nº de Orden	Símbolo Nemotécnico	Descripción	Tipo de Variable	Magnitud
1	ReqElicitados	Requisitos elicitados	Endógena / Nivel	PF
2	ReqEspecificad	Requisitos especificados	Endógena / Nivel	PF
3	ReqValidados	Requisitos validados	Endógena / Nivel	PF
4	ReqInadecuados	Requisitos inadecuados	Endógena / Nivel	PF
5	ErroresAnalisis	Errores de Análisis	Endógena / Nivel	PF
6	Elicitacion	Elicitación de requisitos	Endógena / Flujo	PF
7	Especificacion	Especificación de requisitos	Endógena / Flujo	PF / día
8	Validacion	Validación de requisitos	Endógena / Flujo	PF / día
9	RechazoRequisit	Rechazo de requisitos	Endógena / Flujo	PF / día
10	GestionRequisit	Gestión de requisitos	Endógena / Flujo	PF / día
11	GeneraNvosRequi	Generación de nuevos requisitos	Endógena / Flujo	PF / día
12	GeneraErrAnalis	Generación de errores de análisis	Endógena / Flujo	PF / día
13	FuncionalidadIn	Tamaño inicial del software a ser producido	Endógena / Auxiliar	PF
14	EfzoEspecificac	Esfuerzo asignado a la especificación de requisitos	Endógena / Auxiliar	Empleado / día
15	ProducEspecific	Productividad de especificación	Endógena / Auxiliar	PF / Empleado / día
16	MedidaExacEspec	Medida de la exactitud de la especificación	Endógena / Auxiliar	%

Tabla IV.1. Definición de variables del modelo propuesto (continuación)

Nº de Orden	Símbolo Nemotécnico	Descripción	Tipo de Variable	Magnitud
17	TasaGenerNvsReq	Tasa de Generación de nuevos requisitos	Exógena / Parámetro	% (2)
18	TasaRechazoReq	Tasa de rechazo de requisitos	Exógena / Parámetro	% (10)
19	CteConversionPF	Constante de conversión a Puntos de Función	Exógena / Parámetro	Ctte. (128)
20	LOC	Líneas de Código	Exógena / Parámetro	Ctte.
21	PorcEfzoEspecif	Porcentaje de Esfuerzo de Especificación	Exógena / Parámetro	% (10)
22	CtteMedExac1	Constante de la ecuación de Medida de Exactitud de la Esp.	Exógena / Parámetro	Ctte. (1)
23	CtteMedExac2	Constante de la ecuación de Medida de Exactitud de la Esp.	Exógena / Parámetro	Ctte. (0,02)
24	DiasTrabajoMes	Días de Trabajo al Mes	Exógena / Parámetro	Ctte. (20)
25	NumEmplAnalisis	Número de empleados para el Análisis	Exógena / Parámetro	Ctte. (3,2)

Tabla IV.1. Definición de variables del modelo propuesto (continuación)

IV.1.5. *Construcción del sistema de ecuaciones*

Pasar del diagrama de Forrester al conjunto de ecuaciones, implica considerar tres momentos, que se observan en la Figura IV.16:

J → Instante anterior al actual

K → Instante actual

L → Instante siguiente al actual

Gráficamente:

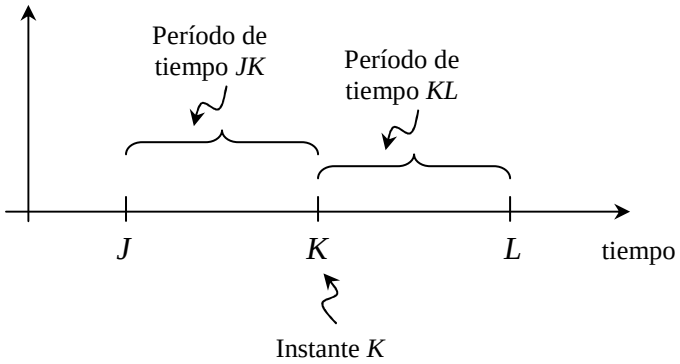


Figura IV.16 Momentos en del Diagrama de Forrester

El sistema siempre se evalúa en el instante *K*, pero se toman como base los resultados de *J* y se proyectan a *L*.

Variables de Nivel:

A continuación se definen las ecuaciones de las variables de nivel que representan la acumulación de flujos (requisitos) y se definen según la forma general:

$N(K) = N(J) + DT * [FE(JK) - FS(JK)]$ donde:

N(K) es el valor del nivel en el instante T

N(J) es valor del nivel en el instante anterior

DT es el intervalo de tiempo elegido

FE(JK) son los flujos de entrada en el período T

FS(JK) son los flujos de salida en el período T

ReqElicitados(K) = ReqElicitados(J) + [Elicitacion(JK) + GestionRequisit(JK) + GeneraNvosRequi(JK) - Especificacion(JK)]DT

ReqEspecificad(K) = ReqEspecificad(J) + [Especificacion(JK) - Validacion(JK) - RechazoRequisit(JK) - GestionRequisit(JK)]DT

ReqValidados(K) = ReqValidados(J) + [Validacion(JK) - TasaDesarroSoft(JK)]DT

ReqInadecuados(K) = ReqInadecuados(J) + [RechazoRequisit(JK)]DT

ErroresAnalisis(K) = ErroresAnalisis (J) + [GeneraErrAnalis(JK)]DT

Variables de Flujo:

Las siguientes ecuaciones corresponden a la definición de las variables de flujo del modelo propuesto, que regulan los cambios de las variables de nivel, se miden en el intervalo de tiempo *KL* que va desde el instante actual al instante siguiente y tienen la siguiente forma:

$F(KL) = N(K) * FN * M(K)$ donde:

F es la variable de flujo

N es el nivel

FN es el flujo normal

M es un multiplicador del flujo normal (se usa para alterar el comportamiento normal de la variable).

Elicitacion(KL) = FuncionalidadIn(K)

*Especificacion(KL) = MIN(ReqElicitados(K), EfzoEspecificac(K) * ProducEspecific(K))*

*Validacion(KL) = ReqEspecificad(K) * (1 - MedidaExacEspec(K) - 0,1)*

*RechazoRequisit(KL) = ReqEspecificad(K) * 0,1*

*GestionRequisit(KL) = ReqEspecificad(K) * MedidaExacEspec(K)*

$$GeneraNvosRequi\ (KL) = \begin{cases} 0 \quad si \quad ReqValidados(K) + ReqInadecuados(K) > \\ \qquad FuncionalidadIn(K) + FuncionalidadIn(K) * 0{,}001 \\ FuncionalidadIn(K) * 0{,}001 \quad en\ caso\ contrario \end{cases}$$

GeneraErrAnalis(KL) = ReqInadecuados(K)/ FuncionalidadIn(K)

Variables Auxiliares:

Las variables auxiliares que se definen a continuación permiten explicar los valores de las variables de flujos y no tienen una fórmula específica.

FuncionalidadIn(K) = LOC / 128

*EfzoEspecificac(K) = EsfuerzoTotal(K) * 0,1*

ProducEspecific(K) = ProducDesarSoft(K) / 128

$$MedidaExacEspec(K) = 1 - \frac{\frac{FuncionalidadIn(K)}{EfzoEspecificac(K) * 20}}{3,2} * 0,02$$

El modelo matemático completo, es decir el sistema de ecuaciones del modelo base más estas ecuaciones definidas precedentemente para el *Modelo* propuesto de la etapa de análisis, está constituido por 28 ecuaciones de nivel, 41 ecuaciones de flujo, 96 ecuaciones de variables auxiliares, 35 parámetros, 27 no linealidades, 4 retardos, para un total de 306 ecuaciones y 348 relaciones entre los elementos.

Formalizado el modelo a nivel del sistema de ecuaciones, ya se está en condiciones de proceder a la construcción de la herramienta de simulación.

El modelo se implementa en EVOLUTION (versión 4.0), herramienta para modelación en Dinámica de Sistemas (DS) en ambiente Windows, desarrollada por el Grupo SIMON de Investigaciones de la Universidad Industrial de Santander de Colombia.

Se selecciona EVOLUTION por ser una herramienta software para modelar y simular con DS, que ya ha ganado reconocimiento nacional e internacional. Además, el grupo de investigación que la desarrolló, se aboca específicamente al apoyo académico, razón por la cual esta herramienta se encuentra disponible sin límites de licencia en sus distintas versiones y con sus correspondientes manuales de usuario [11].

Esta aplicación permite dibujar los distintos tipos de variables que componen el sistema y automáticamente va creando las posibles relaciones entre las mismas, antes de definir las ecuaciones. También tiene la capacidad de vincular grupos de variables como submodelos, para manejar modelos complejos. Ofrece sofisticadas características gráficas y estadísticas para manejar las entradas y mostrar las salidas.

Para simplificar la codificación del modelo en el entorno de la herramienta seleccionada se construyeron por separado:

- Diagramas de flujo-nivel para cada uno de los subsistemas, sectores y subsectores correspondientes a la estructura del modelo base (descripta en el Capítulo III).
- Diagrama de flujo-nivel para el módulo de la etapa de análisis correspondiente al modelo propuesto (descrito en el presente capítulo).

A continuación se muestran, desde la Figura IV.17 a la Figura IV.26, los diagramas de flujo-nivel que constituyen el modelo integrado.

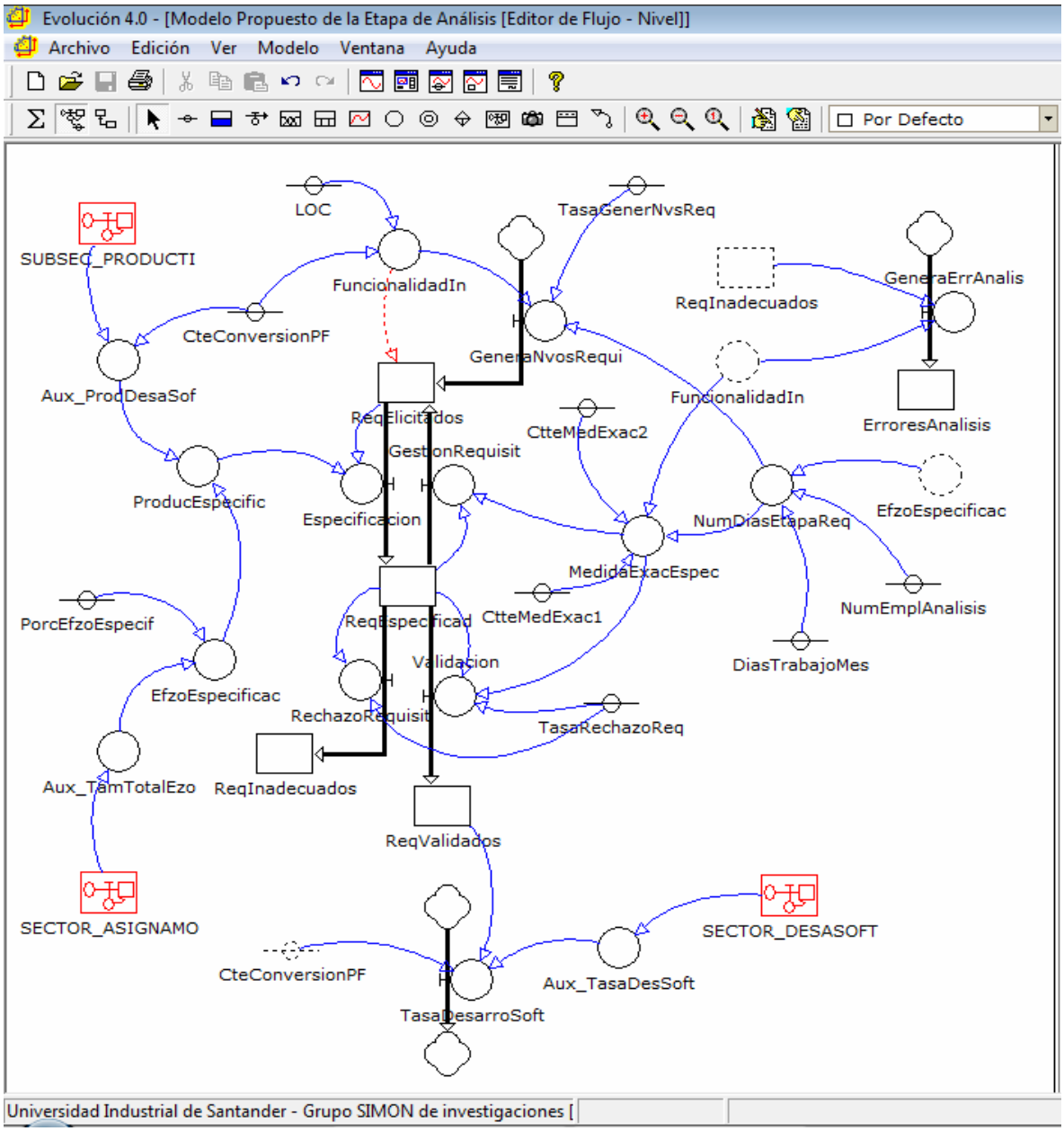


Figura IV.17. Diagrama de Flujo - Nivel del Modelo Propuesto de la Etapa de Análisis

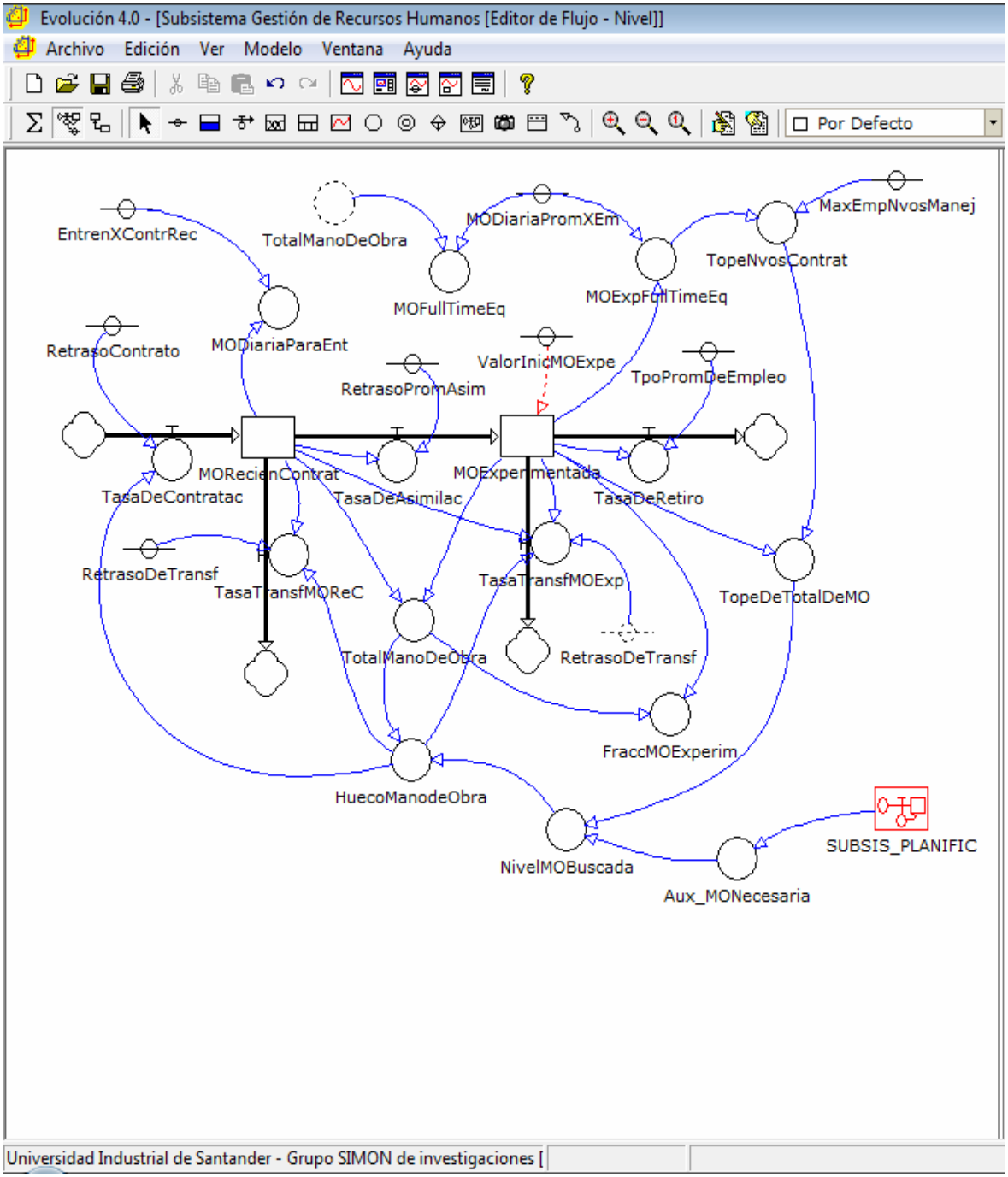


Figura IV.18. Diagrama de Flujo - Nivel del Subsistema "Gestión de Recursos Humanos"

Subsistema "Producción"

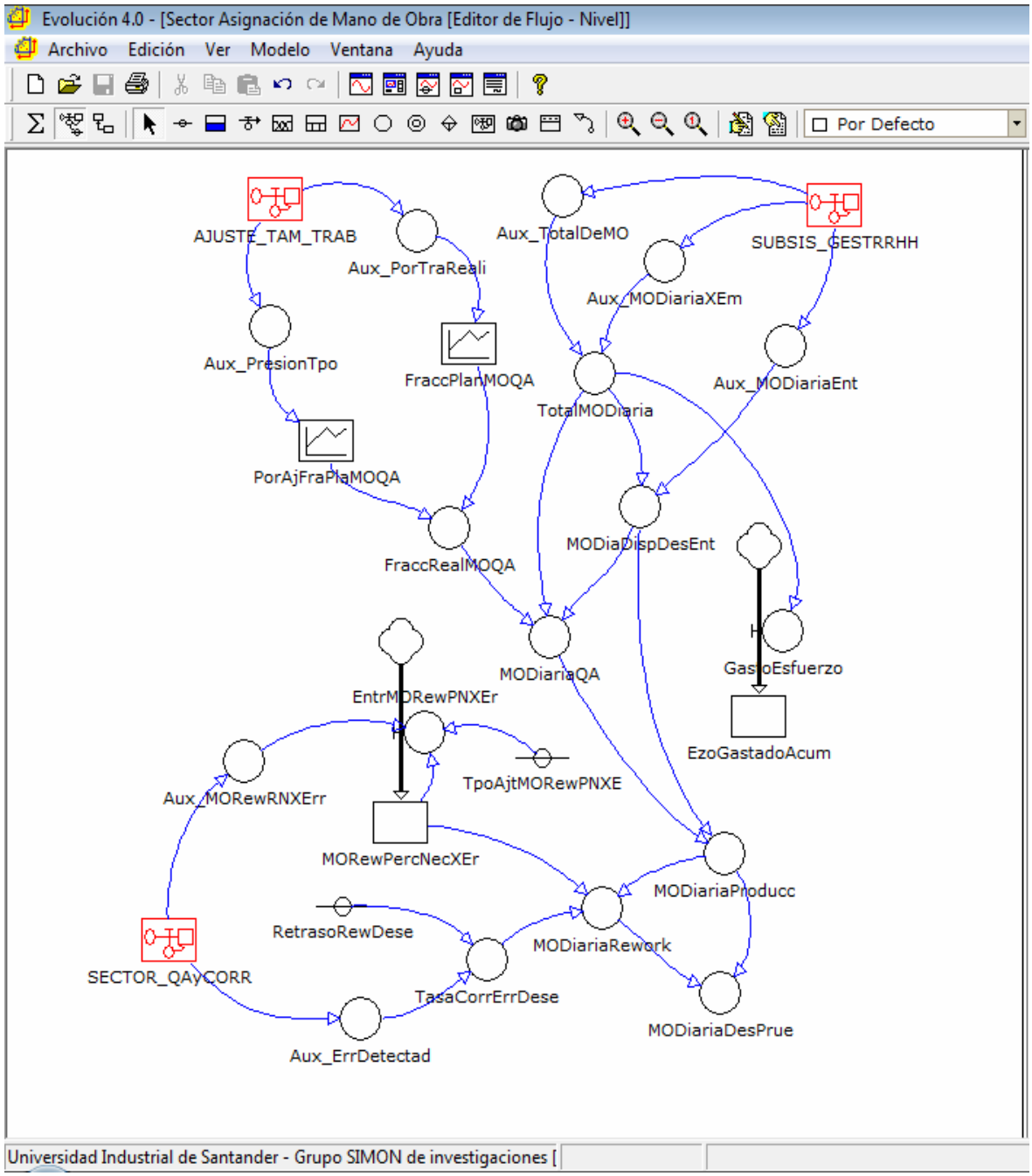


Figura IV.19. Diagrama de Flujo - Nivel del Sector "Asignación de Mano de Obra"

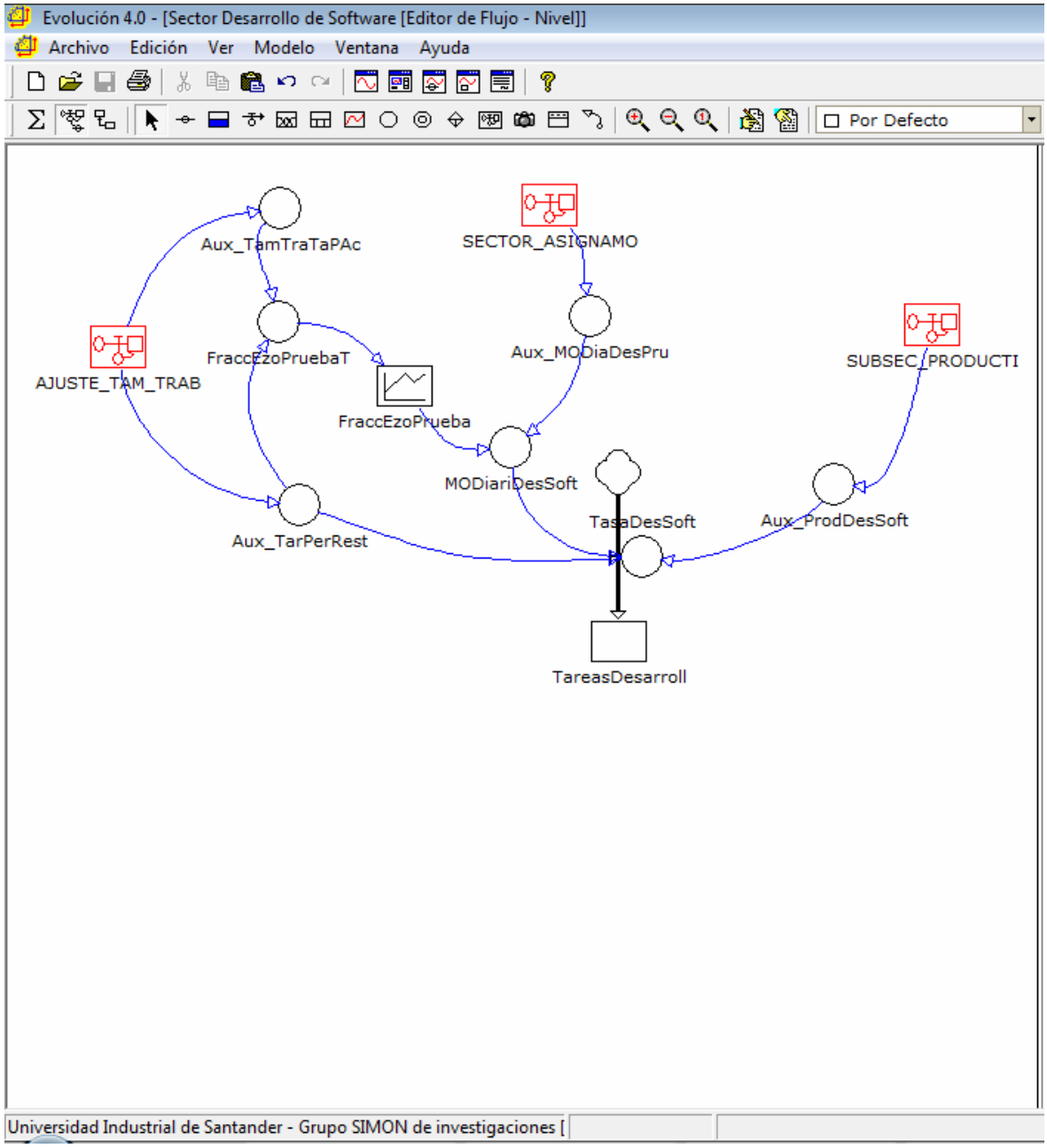


Figura IV.20. Diagrama de Flujo - Nivel del Sector "Desarrollo de Software"

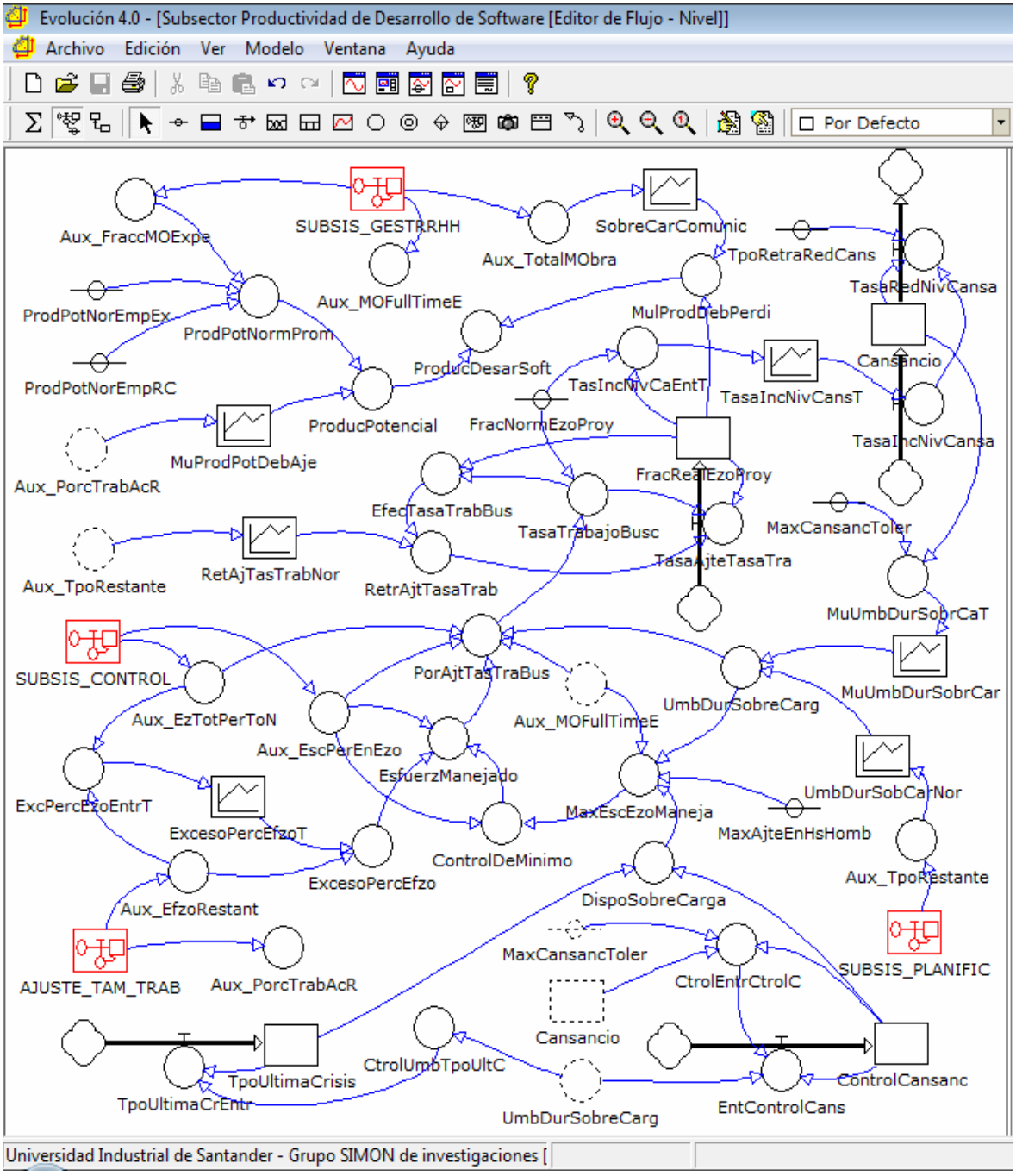


Figura IV.21. Diagrama de Flujo - Nivel del Subsector "Productividad de Desarrollo de Software"

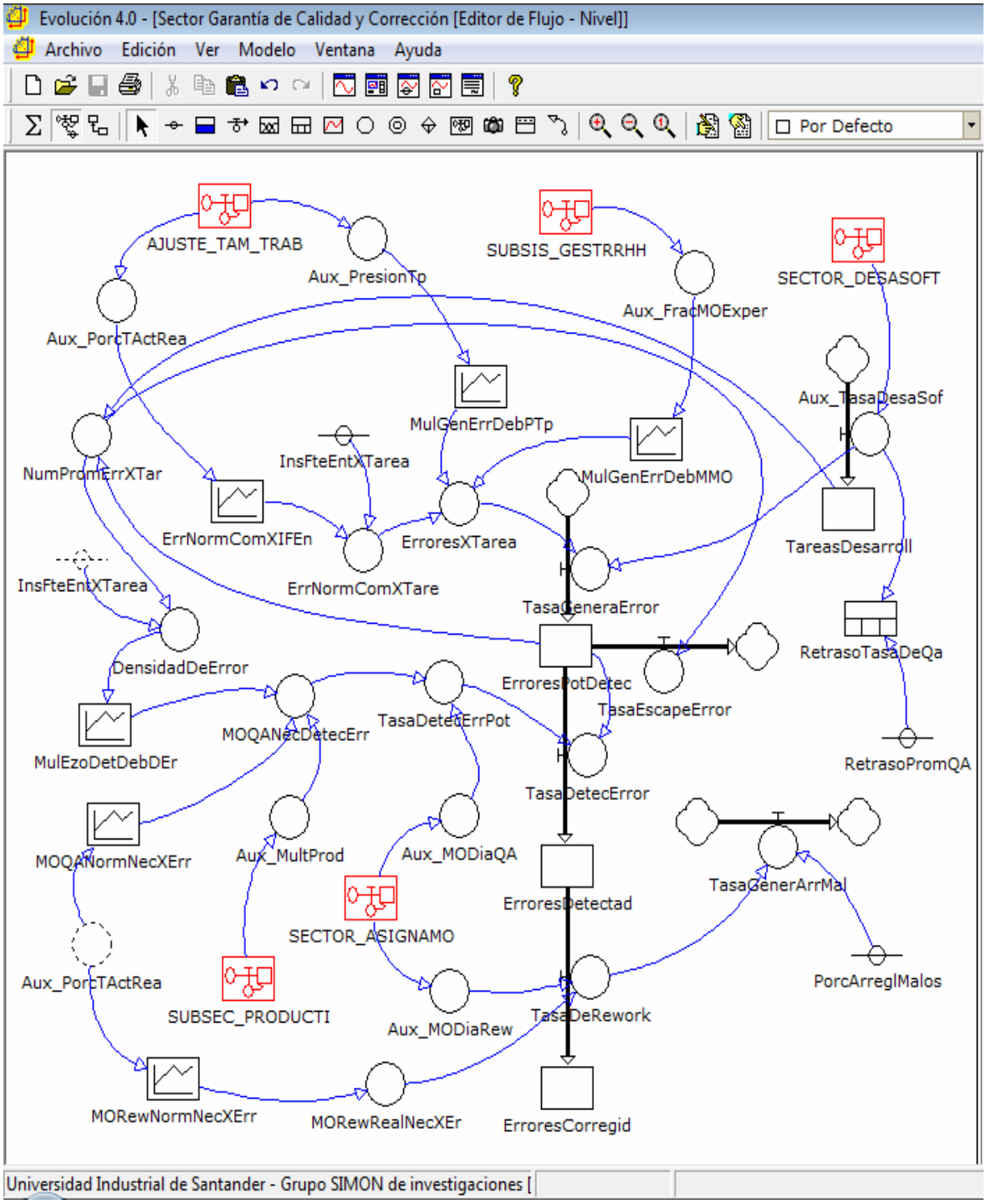


Figura IV.22. Diagrama de Flujo - Nivel del Sector "Garantía de Calidad y Corrección"

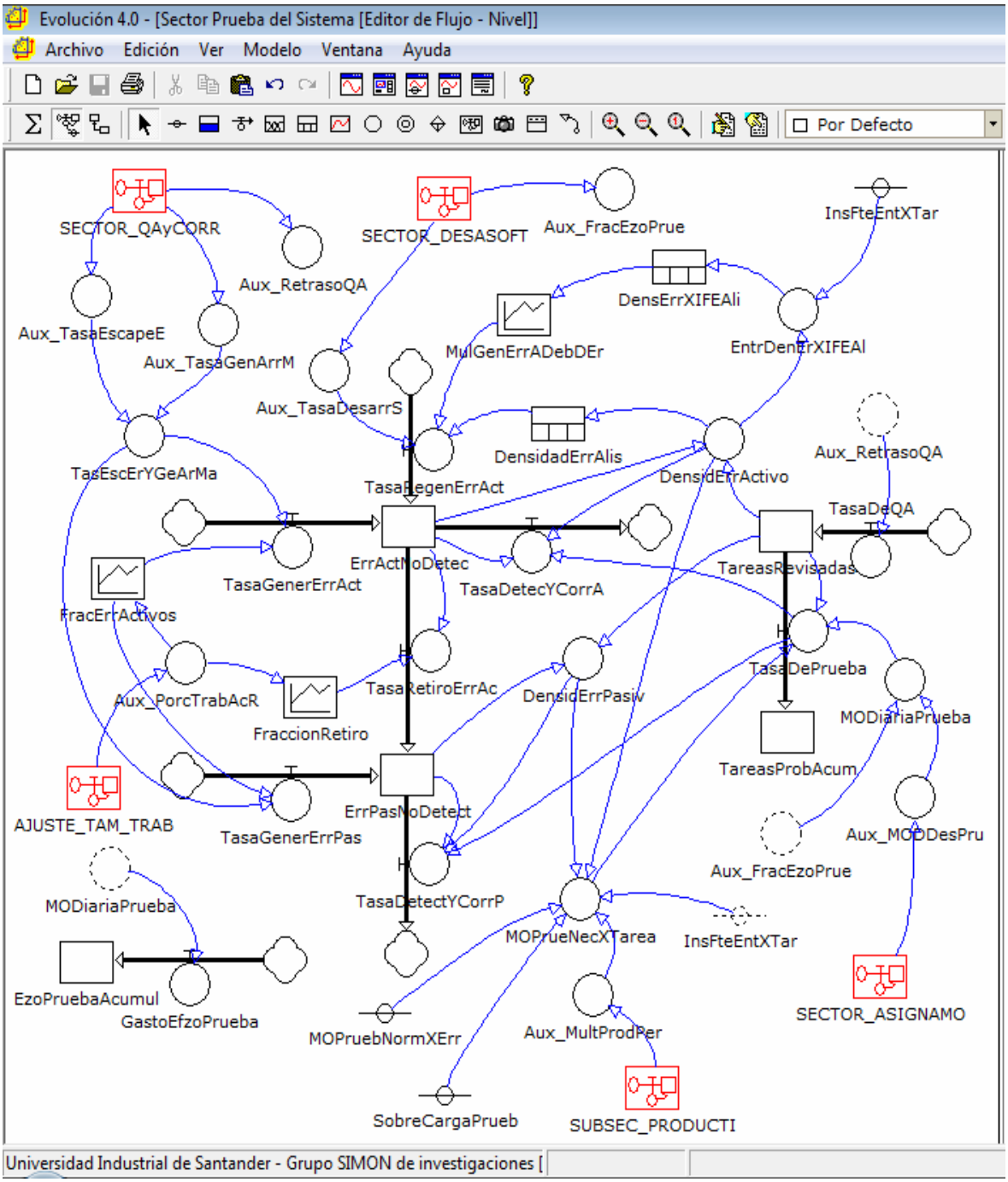


Figura IV.23. Diagrama de Flujo - Nivel del Sector "Prueba del Sistema"

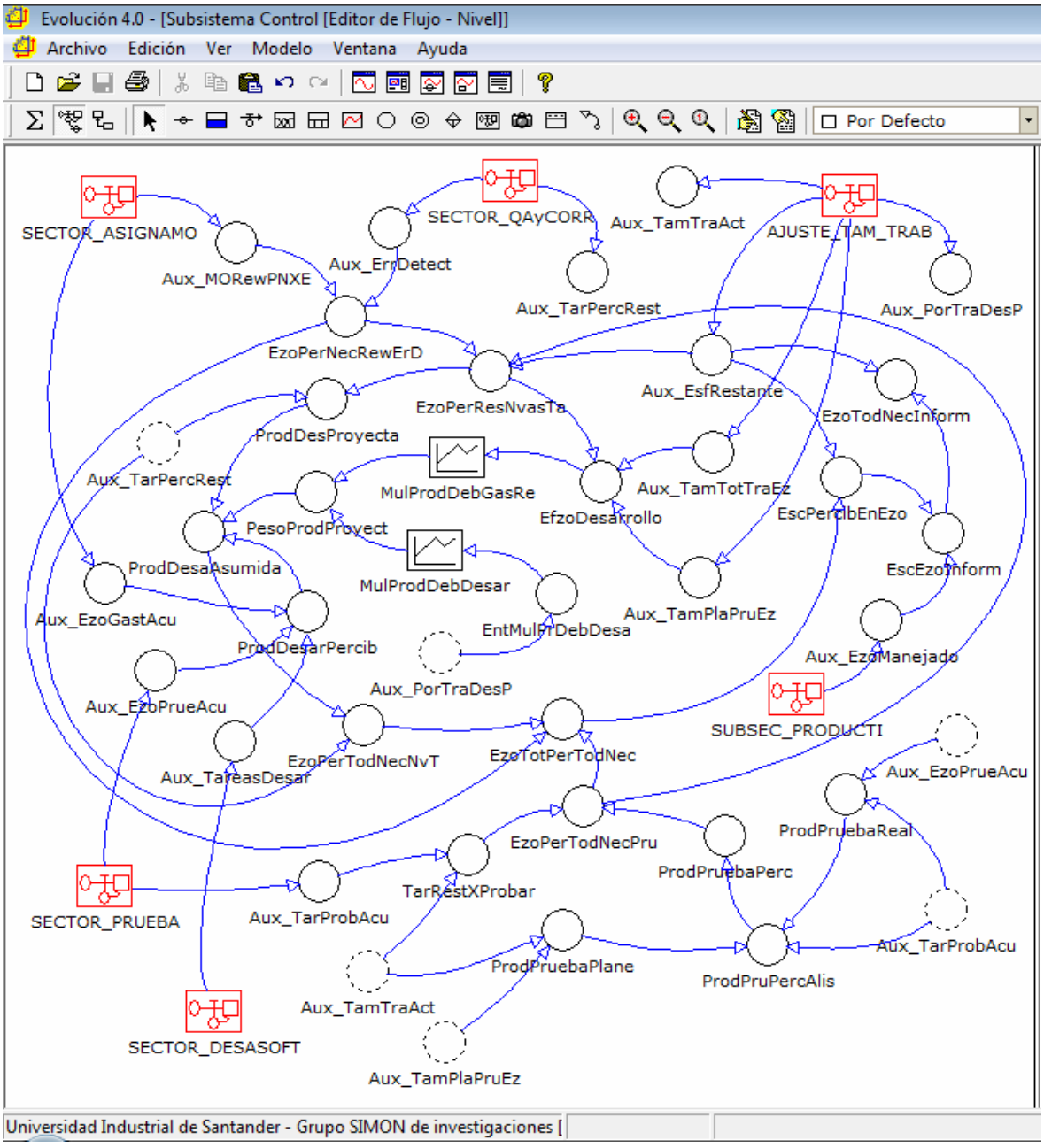


Figura IV.24. Diagrama de Flujo - Nivel del Subsistema "Control"

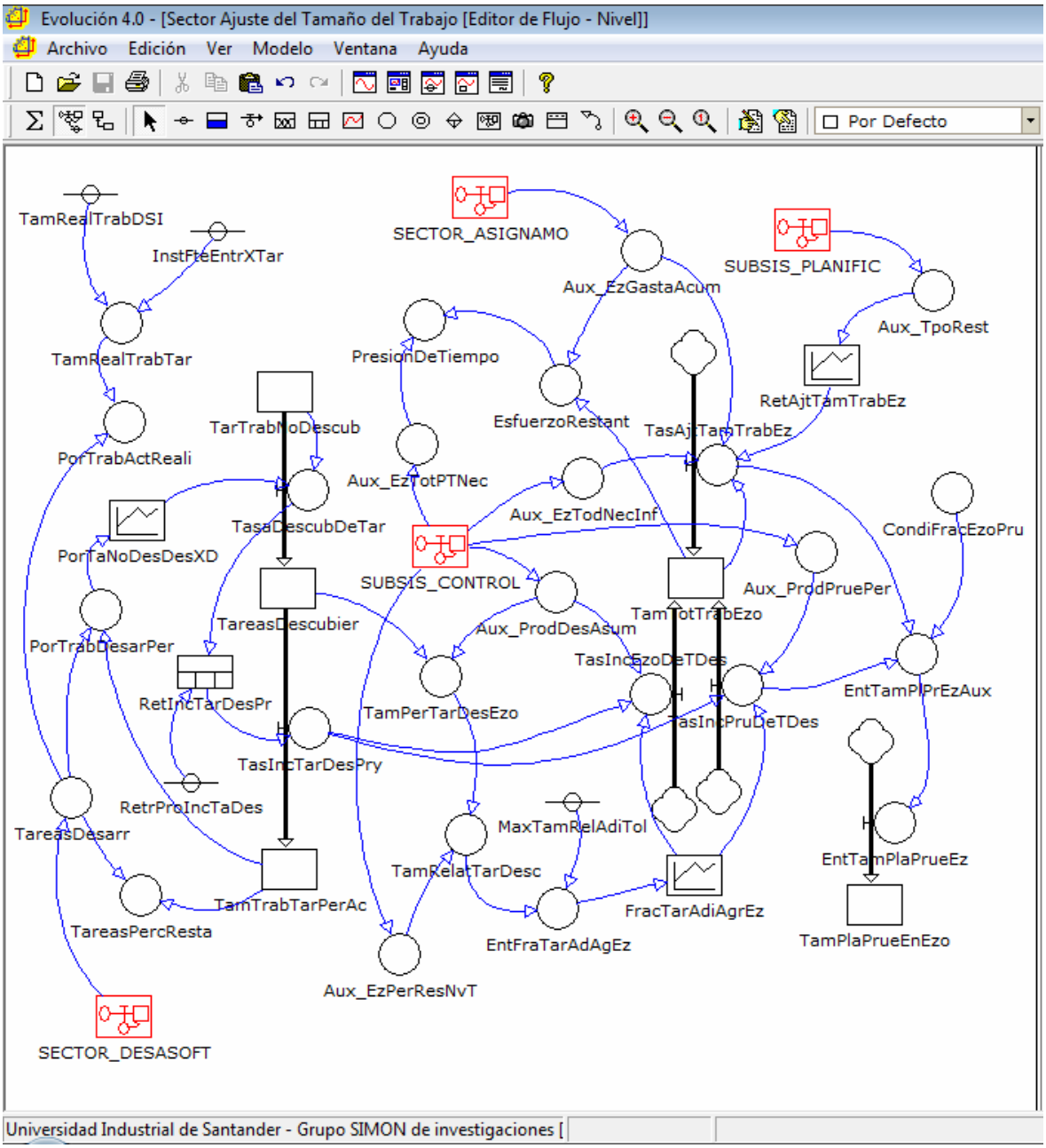


Figura IV.25. Diagrama de Flujo - Nivel del Sector "Ajuste del Tamaño del Trabajo"

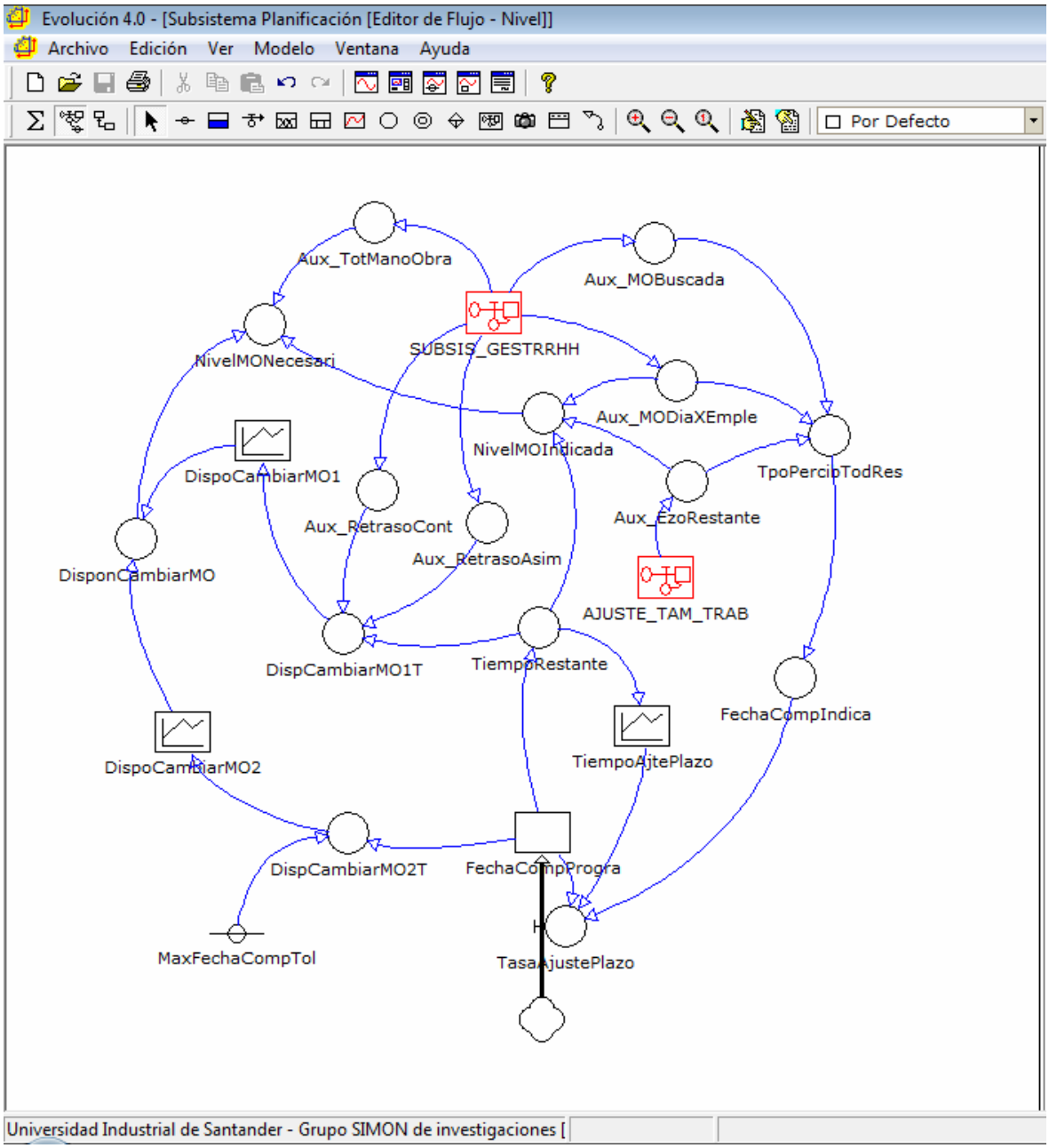


Figura IV.26. Diagrama de Flujo - Nivel del Subsistema "Planificación"

Si bien cada uno de los subsistemas, sectores y subsectores se presentaron por separado para reducir la complejidad visual, a efectos de realizar las sucesivas ejecuciones de simulación para completar las subsiguientes etapas de la metodología de Dinámica de Sistemas, cada uno de estos diagramas de flujo-nivel se vinculan a través del elemento "Submodelo" provisto por el software Evolution.

Este elemento de vinculación se representa con: 

y es el que se puede apreciar, por ejemplo, en el diagrama de flujo-nivel del modelo propuesto de la etapa de análisis y que se comporta a modo de interfase con los diagramas de flujo-nivel del modelo base, como se muestra en la Figura IV.17.

Dado que el modelo base ya ha sido probado por quienes lo desarrollaron, las siguientes etapas de Calibrado y Análisis de Sensibilidad, sólo se realizarán para parámetros del modelo propuesto de la etapa de análisis. Luego en el Capítulo V, al probar el modelo completo con la aplicación al caso de estudio, se podrán realizar los ajustes que resulten necesarios en relación a los parámetros compartidos (entre el modelo base y el propuesto) en forma directa, o bien, parámetros que tengan incidencia sobre otras variables que sean compartidas.

IV.1.6. *Calibrado*

El calibrado es como una "puesta a punto" del modelo que se realiza por medio de la inclusión de los resultados obtenidos por los expertos en relación al fenómeno en estudio. Esta recopilación de información se suministra al modelo por medio de variables exógenas, variables auxiliares y tablas. Básicamente, el calibrado, consiste en determinar el valor que toman los parámetros definidos para el modelo (valores fijos para la simulación) y para ello, lo más común es utilizar la estadística para determinar valores promedios y tasas que se incluirán en el modelo.

La información para inicializar los parámetros del modelo propuesto, así como para dirigir la evolución del mismo, se ha derivado parcialmente de investigaciones previas [1] [8] [19].

La distribución de estos valores de los parámetros de la etapa de análisis, refleja el escenario "por defecto" que se ha supuesto para una primera simulación y se detallan a continuación en la Tabla IV.2.

Símbolo Nemotécnico del Parámetro	**Descripción**	**Valor**
TasaGenerNvsReq	Tasa de Generación de nuevos requisitos	2 %
TasaRechazoReq	Tasa de rechazo de requisitos	10 %
CteConversionPF	Constante de conversión a Puntos de Función	128
LOC	Líneas de Código	18000 (*)
PorcEfzoEspecif	Porcentaje de Esfuerzo de Especificación	20 %
CtteMedExac1	Constante de la ecuación de Medida de Exactitud de la Esp.	1
CtteMedExac2	Constante de la ecuación de Medida de Exactitud de la Esp.	0,02
DiasTrabajoMes	Días de Trabajo al Mes	20
NumEmplAnalisis	Número de empleados para el Análisis	3,2
(*) Valor hipotético que se encuadra dentro del intervalo de tamaño del proyecto de mediana envergadura. Para la prueba del modelo completo se definirá su valor desde el caso de estudio seleccionado.		

Tabla IV.2. Valores de los parámetros del modelo de la Etapa de Análisis Propuesto

Calibrar el modelo también implica establecer los valores de sus parámetros con el objeto de obtener resultados acordes a lo esperado o, dicho de otra manera, que cuando se lo simule produzca como resultado diversos estados del sistema semejantes a los del comportamiento real. En este caso, el modelo fue preparado para representar un período de tiempo ya transcurrido (análisis post mortem) por lo que esta etapa es un primer paso para alcanzar la validez del modelo. Luego, los resultados de sucesivas simulaciones, para los distintos valores de parámetros del proceso de calibración, serán comparados con los

datos reales disponibles del período de tiempo considerado para analizar que tan coherentes son con la realidad.

El modelo propuesto es un *modelo determinista puro*, ya que los valores de los parámetros de entrada son valores simples (por ejemplo, el número de empleados para el análisis es 3,2) y no valores aleatorios que siguen una distribución de probabilidad concreta como ocurre en un *modelo estocástico*. En un modelo determinista puro, para cada conjunto de parámetros de entrada se requiere únicamente una simulación.

Para los valores de parámetros tabulados precedentemente se realizó la primera simulación.

Para iniciar una corrida de simulación se deben definir: las trayectorias de simulación, las condiciones de simulación y el escenario asociado.

- *Trayectorias de Simulación*: Esta definida por todas las variables que componen el sector del Modelo de Análisis propuesto.

La pantalla de la Figura IV.27 muestra la ventana "Definir Trayectorias" de la herramienta EVOLUTION, en la que se seleccionan, de todas las variables que componen el modelo integrado, aquellas que forman parte del módulo de la etapa de análisis para analizar su evolución en el período de tiempo considerado.

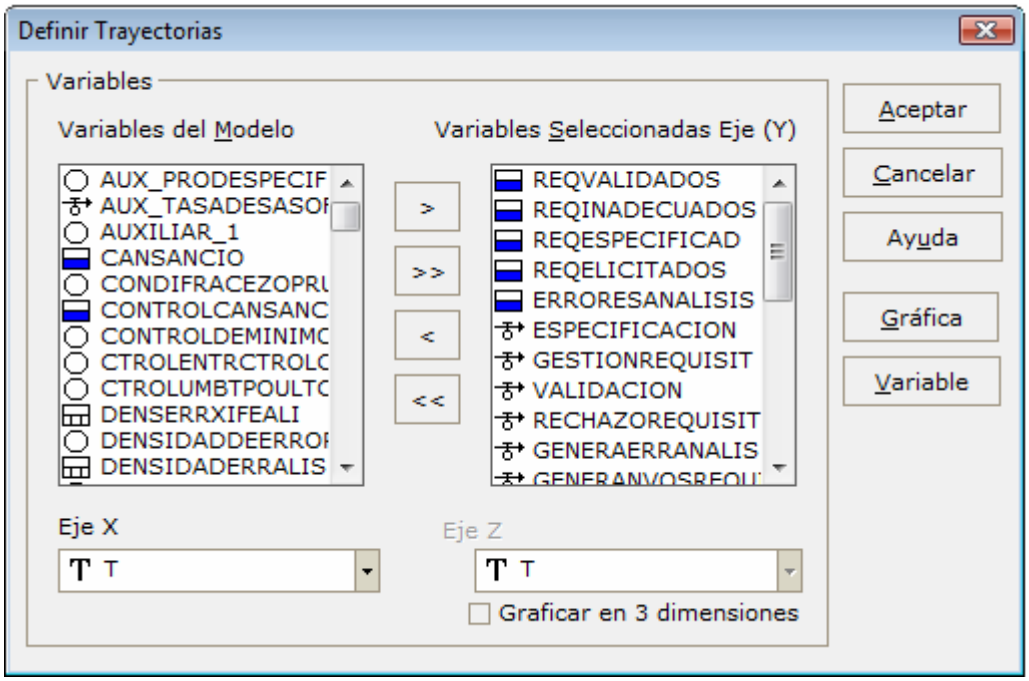


Figura IV.27. Ventana de definición de trayectorias para las variables del Modelo de la Etapa de Análisis Propuesto

• *Condiciones de Simulación*: Implica establecer los tiempos de simulación inicial y final, como así también la selección del método de integración para evaluar las ecuaciones diferenciales que componen el modelo matemático.

Tiempos de Simulación:

Tiempo Inicial = 0

Tiempo Final = 100

Método de Integración: Euler

Estas selecciones se pueden visualizar en la pantalla de la Figura IV.28.

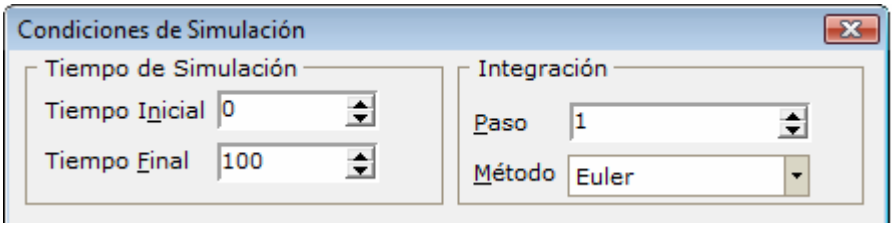


Figura IV.28. Ventana de selección de las condiciones de simulación

• *Escenario asociado*: Es el escenario denominado "*por defecto*" y lo constituye el espacio en el que se disponen los elementos del modelo integrado (niveles, flujos, auxiliares, parámetros, submodelos y tablas, con sus respectivos valores), y las relaciones entre ellos.

La pantalla capturada en la Figura IV.29. indica el escenario asociado al modelo extendido.

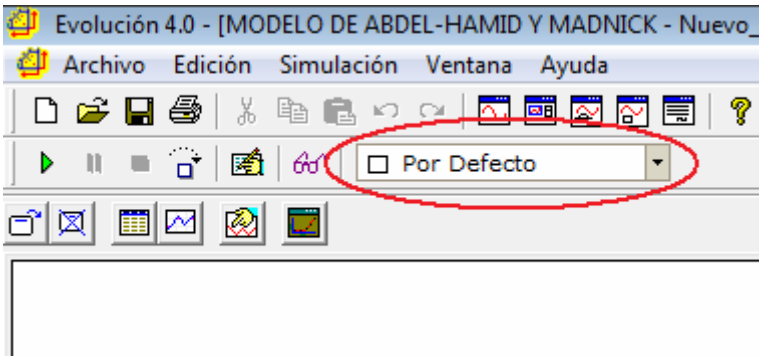


Figura IV.29. Escenario asociado al modelo extendido

Para esta primera simulación de 100 iteraciones, se tomó un valor hipotético de LOC = 115000.

Bajo los valores de parámetros indicados en la Tabla IV.2, los resultados de simulación obtenidos para las variables del modelo de la etapa de análisis, muestran los comportamientos, que se detallan a continuación. Los mismos se analizan como tendencias generales. La evolución temporal de cada variable se muestra en un sistema de ejes cartesianos, donde el eje "*x*" representa la variable en cuestión medida en PF (Puntos de Función) que es la unidad de medida del modelo propuesto de la etapa de análisis. El eje "*y*" representa el tiempo, con tope de 100 días, que se corresponde con las 100 iteraciones.

- ***Niveles:***

- Requisitos Elicitados (ReqElicitados)

Para este nivel, es comportamiento esperado que los requisitos inicialmente elicitados al comienzo del proyecto, disminuyan a medida que avanza el desarrollo en el tiempo. Esto se debe a que la cantidad de requisitos especificados, que es lo que vacía el nivel de requisitos elicitados, en cada iteración es mayor que la fracción de la funcionalidad que determina la generación de nuevos requisitos que aumenta el nivel de requisitos elicitados.

La Figura IV.30. muestra la reproducción del comportamiento descripto para la variable de nivel "*Requisitos Elicitados*", con la herramienta de simulación.

- Requisitos Especificados (ReqEspecificad)

Los requisitos especificados dependen de la productividad de especificación, y ésta del esfuerzo de especificación y de la productividad de desarrollo. Ésta última se toma del modelo base. El valor inicial para este nivel es 0, bruscamente crece a 33,73 requisitos en la primera iteración para luego descender gradualmente con el desarrollo. Este es un comportamiento esperado, ya que cómo lo mencionan los autores del modelo base, a medida que progresa el desarrollo y de acuerdo a la percepción que se tenga del progreso, entran en juego cadenas causa-efecto entre variables: aumenta presión, aumenta la contratación y por lo tanto las vías de comunicación, lo que lleva a disminuir la productividad. También aumenta el cansancio y por lo tanto la generación de error disminuyendo las actividades de producción para desviar a la corrección. Por todo ello, es de esperar que la productividad luego comience a disminuir.

Este comportamiento de la variable "*Requisitos Especificados*" se puede visualizar en la Figura IV.31.

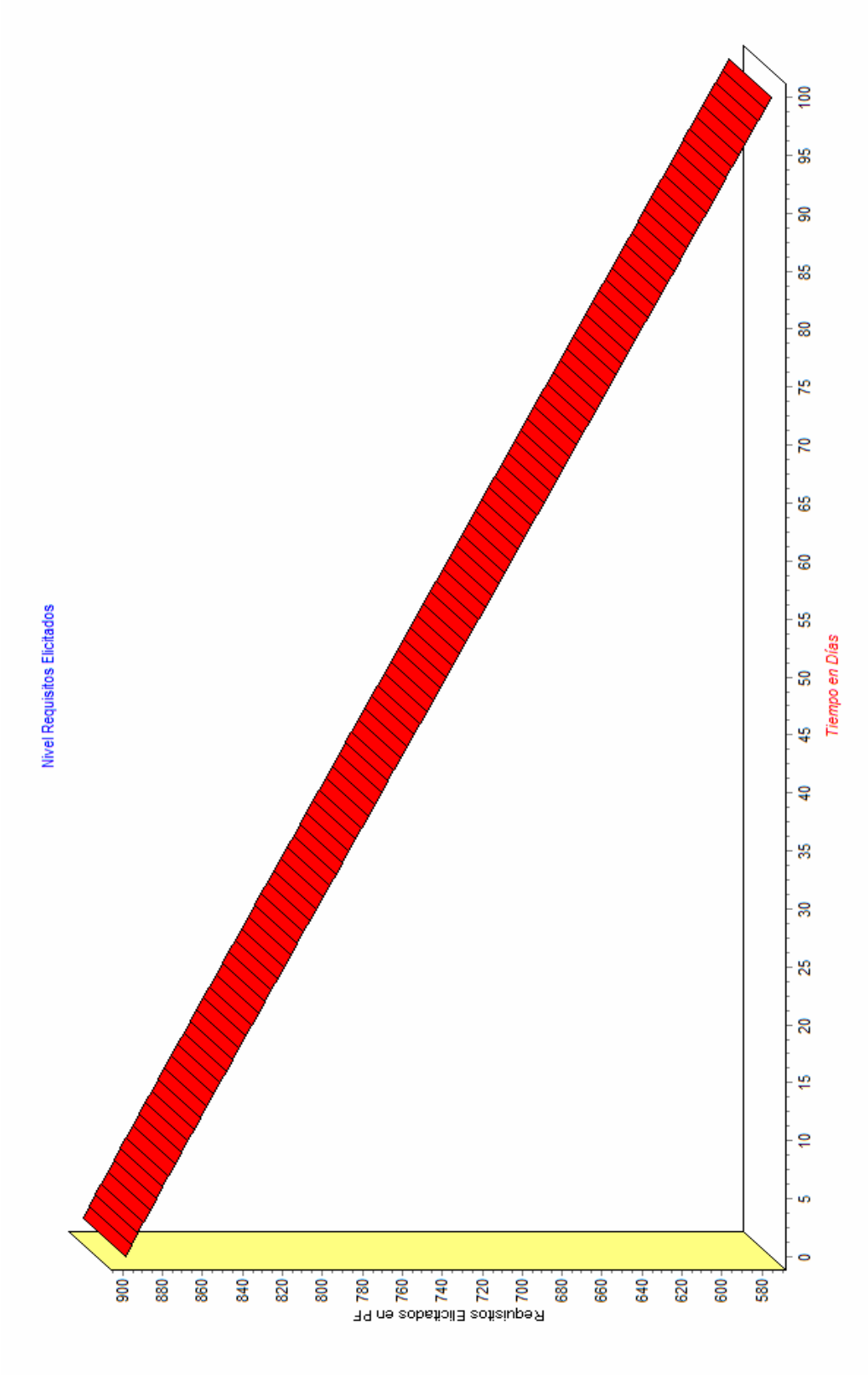


Figura IV.30. Evolución temporal de la variable de nivel “Requisitos Elicitados”

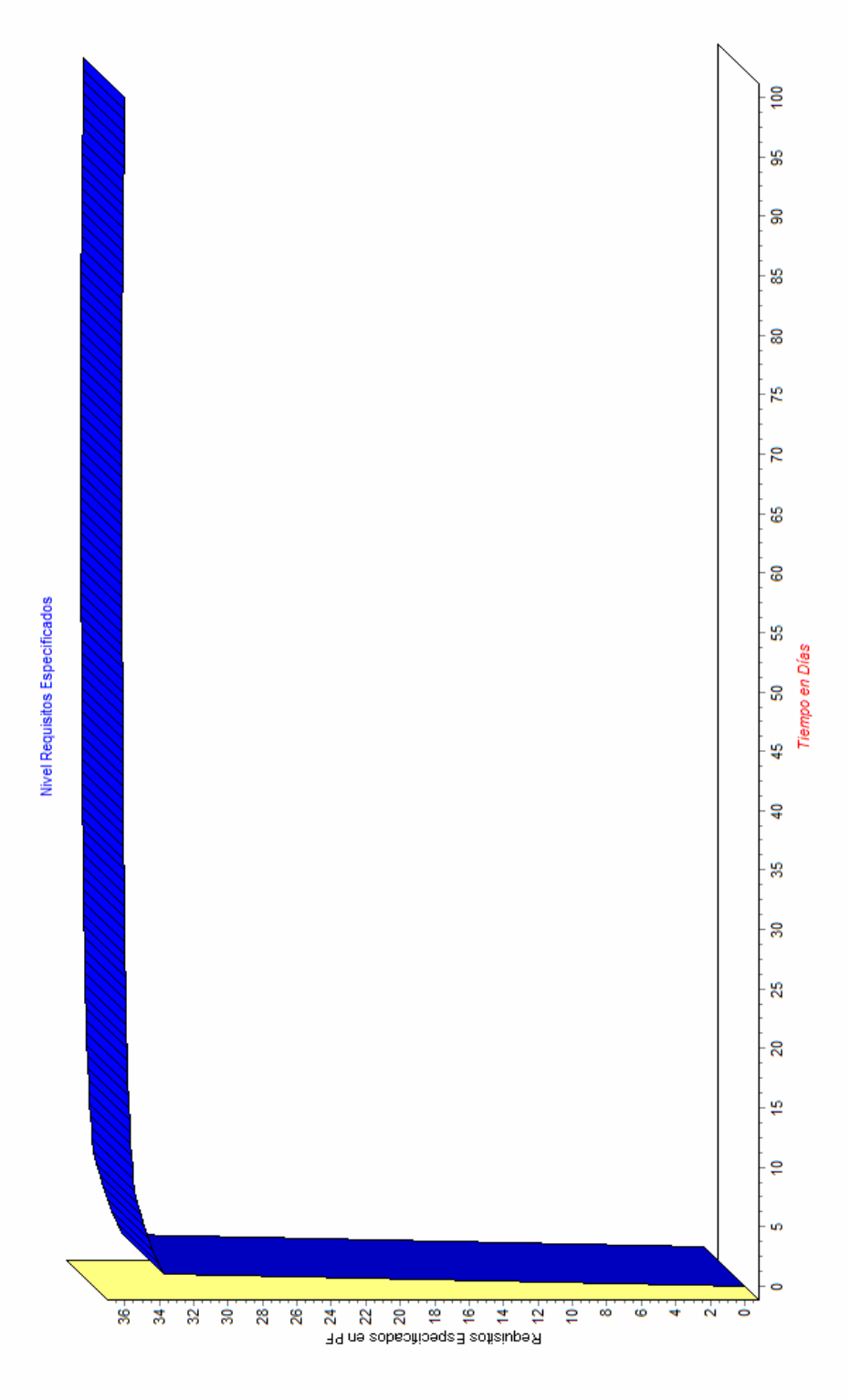


Figura IV.31. Trayectoria de la variable de nivel “Requisitos Especificados”

- Requisitos Inadecuados (ReqInadecuados)

Es de esperar el crecimiento en el tiempo de esta variable, como lo muestra la Figura IV.32, ya que mientras más se extienda el proyecto, más tiempo hay para que surjan nuevos requisitos ya que es más probable que el entorno y/o los usuarios cambien. Esto hace más probable que cambien los requisitos iniciales, que surjan errores al efectuar los cambios y que otros requisitos pasen a ser obsoletos e inadecuados.

A los efectos del presente modelo, este nivel se utiliza puramente para acumulación.

- Requisitos Validados (ReqValidados)

Al igual que ocurre con los requisitos inadecuados, la tendencia general, aunque en distinta proporción, es que a medida que el proyecto progresa, aumenta el número de requisito validados, lo cual se da por el propio trabajo de análisis.

En la Figura IV.33 se puede visualizar la trayectoria que sigue la variable de nivel "*Requisitos Validados*".

- Errores de Análisis (ErroresAnalisis)

Dado que la generación de errores de análisis esta en función de los requisitos inadecuados, el comportamiento de este nivel muestra también una tendencia de aumento a medida que se avanza en el desarrollo. Este comportamiento se muestra en la Figura IV.34.

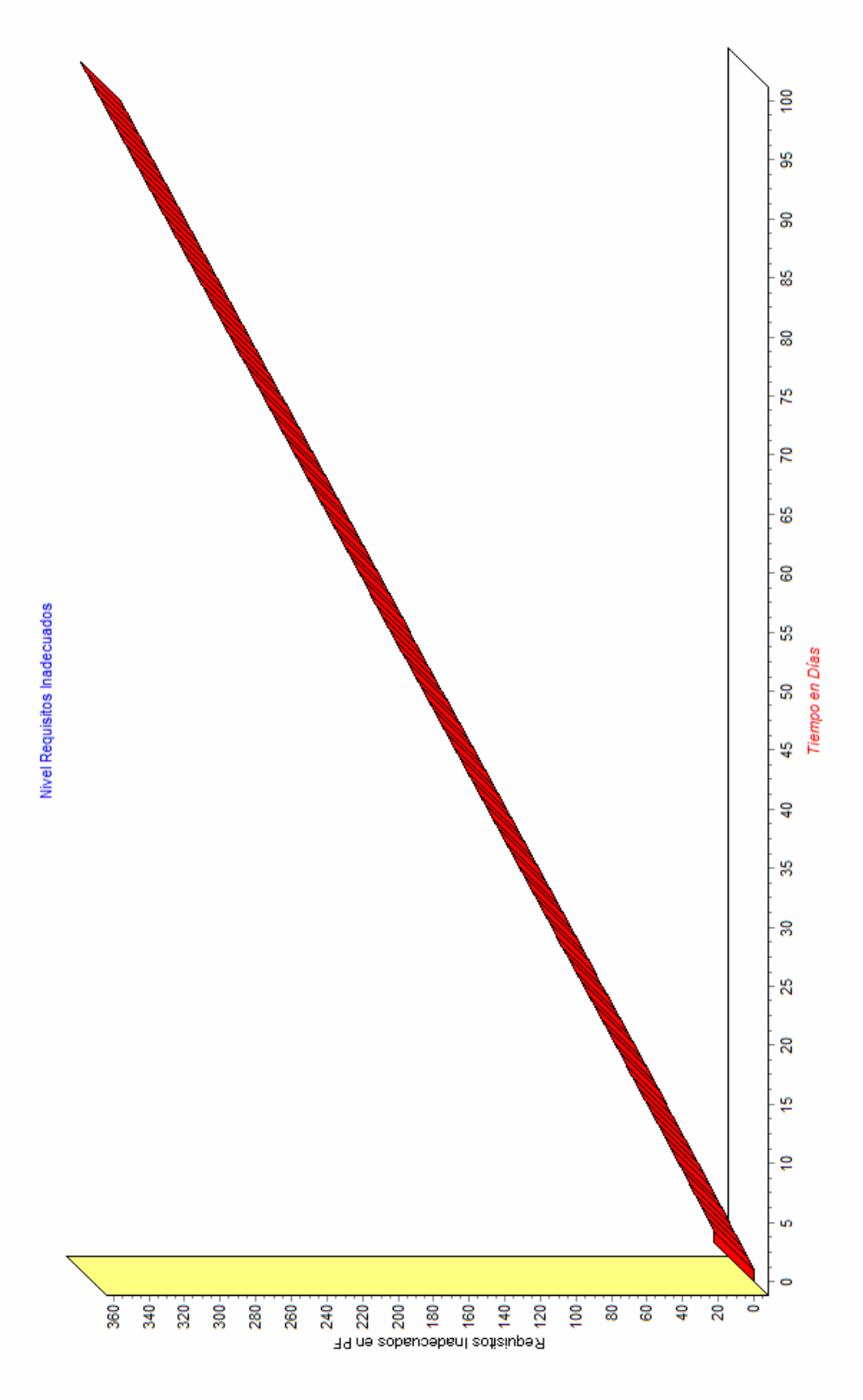


Figura IV.32. Comportamiento de la variable de nivel “Requisitos Inadecuados”

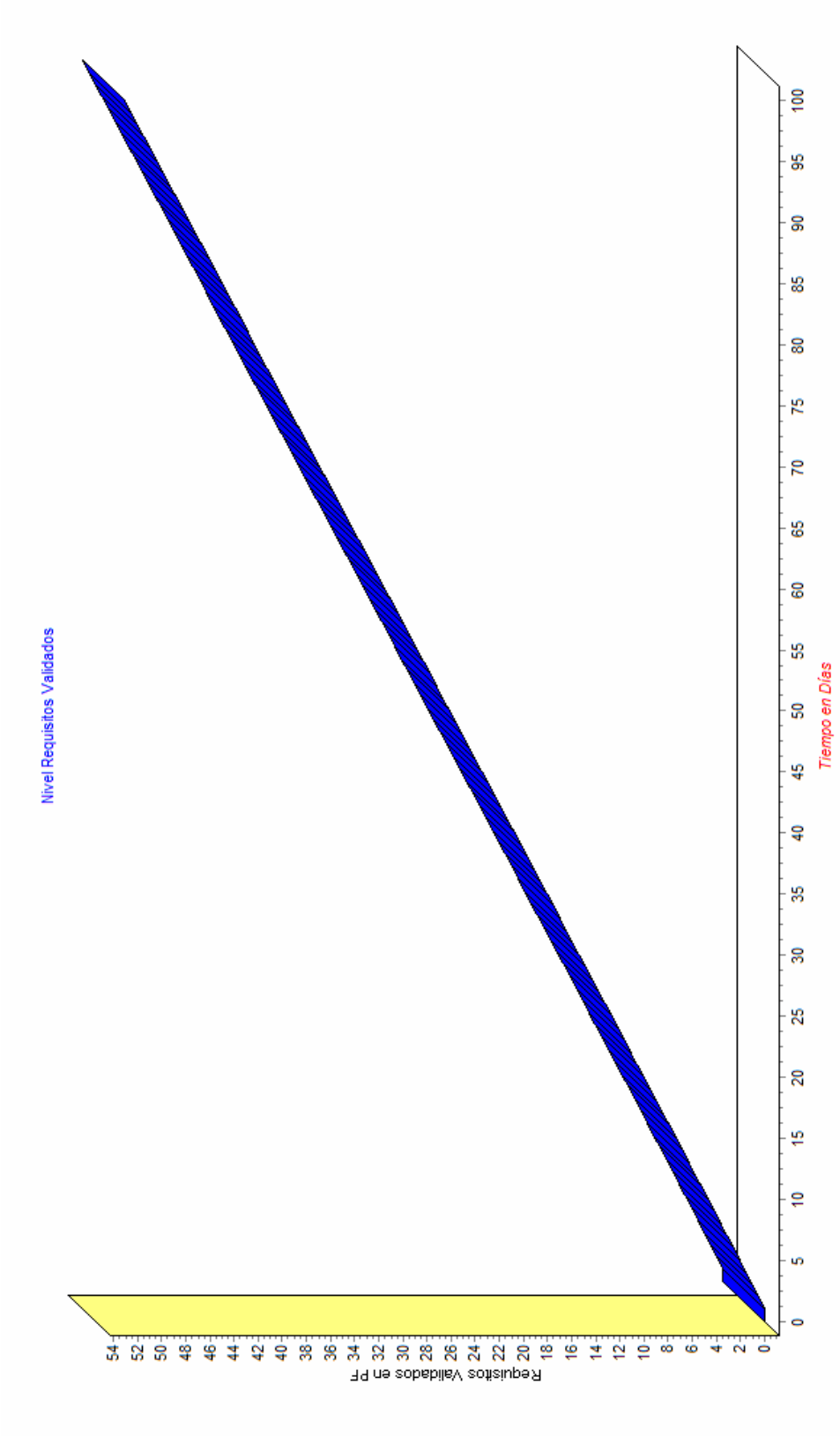


Figura IV.33. Trayectoria de la variable de nivel “Requisitos Validados”

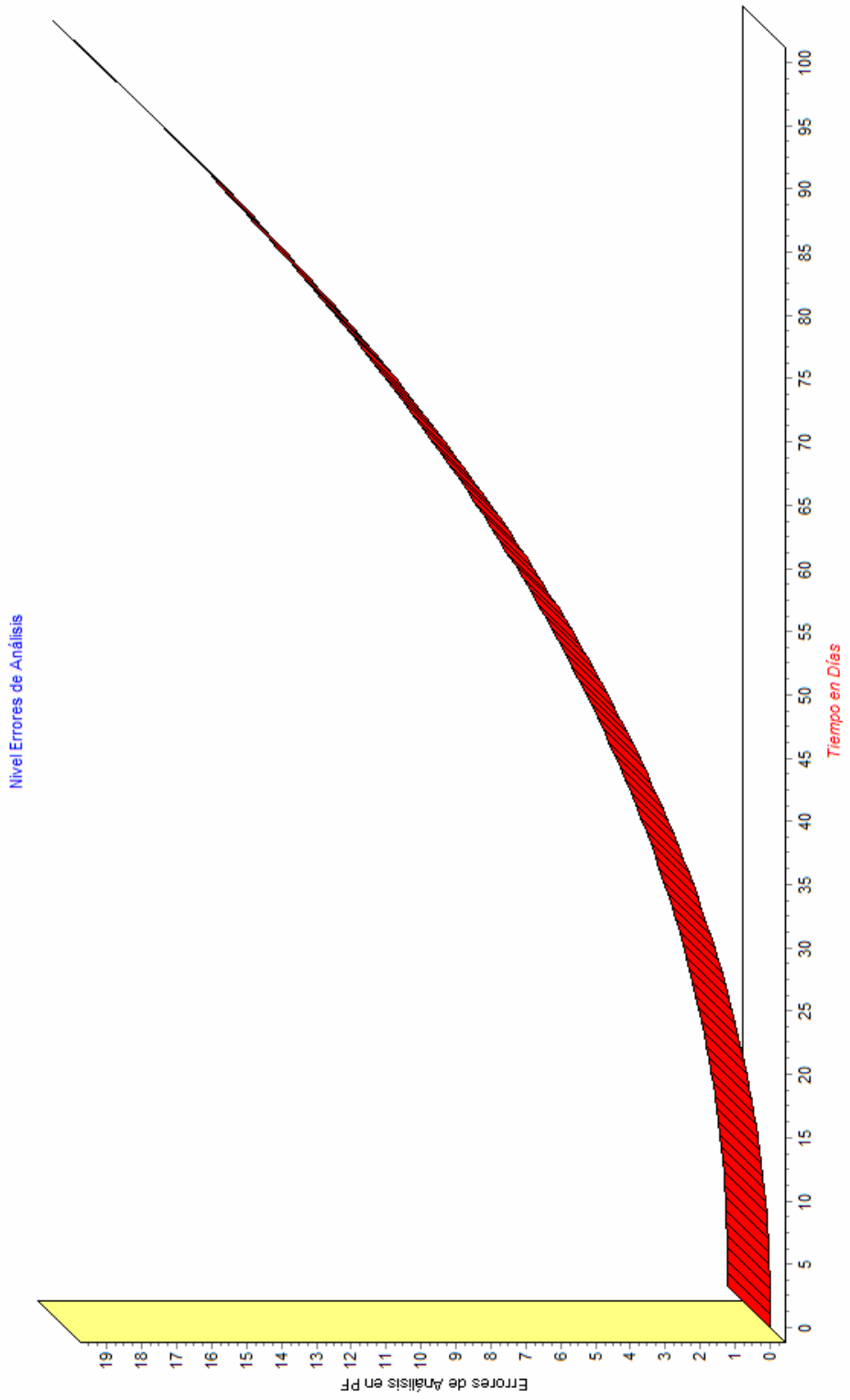


Figura IV.34. Evolución temporal de la variable de nivel “Errores de Análisis”

- ***Flujos:***

- Generación de Nuevos Requisitos (GeneraNvosRequi)

Temiendo en cuenta que según la variable "*NumDiasEtapaReq*", esta etapa dura aproximadamente 156 días, se realizó la simulación para 160 iteraciones y se puede observar que en la iteración 141, se detiene la generación de nuevos requisitos, ya que como se definió el modelo, esto ocurre 15 días antes de finalizar la etapa. Luego, la salida generada por el modelo, es la esperada, como se puede visualizar en la Figura IV.35.

Iterac.	X:T	GENERANVOSREQUI
126	125	0.8984375
127	126	0.8984375
128	127	0.8984375
129	128	0.8984375
130	129	0.8984375
131	130	0.8984375
132	131	0.8984375
133	132	0.8984375
134	133	0.8984375
135	134	0.8984375
136	135	0.8984375
137	136	0.8984375
138	137	0.8984375
139	138	0.8984375
140	139	0.8984375
141	140	0.8984375
142	141	0.8984375
143	142	0
144	143	0
145	144	0
146	145	0
147	146	0
148	147	0
149	148	0
150	149	0
151	150	0
152	151	0
153	152	0
154	153	0
155	154	0
156	155	0
157	156	0
158	157	0
159	158	0
160	159	0
161	160	0

Figura IV.35. Comportamiento de la variable de Flujo "Generación de Nuevos Requisitos"

- Especificación (Especificacion)

El flujo "*Especificación*" esta definido por el mínimo entre la productividad de especificación y requisitos elicitados. Como muestra la Figura IV.36, su valor aumenta al principio del proyecto, pero luego disminuye por las pérdidas en la productividad, como consecuencia de la presión; del aumento de la contratación; de las líneas de comunicación; aumento de la generación de errores y por lo tanto de las tareas de rework y prueba. Esto es tal como se explicara en el comportamiento del nivel de requisitos especificados cuya fluctuación está determinada por este flujo.

- Gestión de Requisitos (GestionRequisit)

El flujo gestión de requisitos definido por la variable "*GestiónRequisit*", es una proporción del nivel de requisitos especificados, determinada por la variable auxiliar "*MedidaExacEspec*" que mide el grado de exactitud de la especificación. Como ésta última toma un valor constante, para la 100 iteraciones, por estar definida por valores de parámetros, el valor del flujo que se esta considerando variará entonces de acuerdo al valor del nivel "*ReqEspecificad*", de modo que el comportamiento de "*GestiónRequisit*" sigue en forma análoga el del nivel mencionado. Es decir el valor del flujo gestión de requisitos (fracción de requisitos especificados, determinada por "*MedidaExacEspec*"), crece al principio del desarrollo y disminuye luego, como es de esperar, ya que al avanzar el proyecto y finalizanda la etapa de requisitos, se detiene la generación de nuevos requisitos, otros se van descartando, otros se van validando y el ciclo de gestión va purificando cada vez menos cantidad de requisitos pendientes para analizar y especificar. Este comportamiento está representado en la Figura IV.37.

- Rechazo de Requisitos y Validación (RechazoRequisit y Validacion)

Estos dos últimos flujos y al igual que el anterior, muestran una tendencia de crecimiento al principio para disminuir luego. Esto porque tienen dependencia, aunque en distintas proporciones, del mismo nivel (requisitos especificados). En el primer flujo la proporción está determinada por la tasa de rechazo de requisitos y, en el segundo flujo, está determinada por la tasa de rechazo de requisitos y la medida de exactitud de las especificaciones. Las Figuras IV.38 y IV.39 muestran las trayectorias para estos dos flujos analizados.

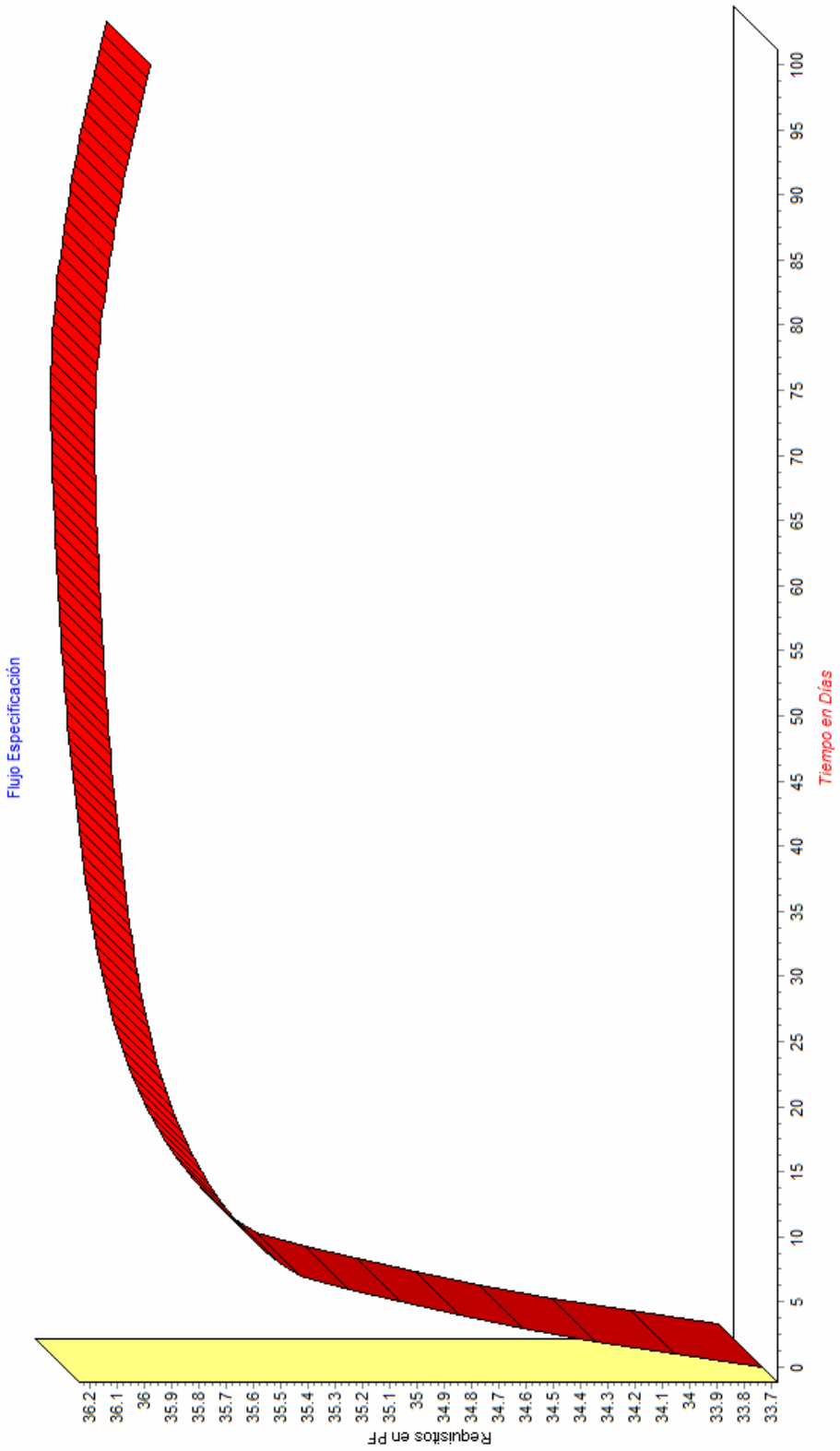


Figura IV.36. Trayectoria de la variable de flujo "Especificación"

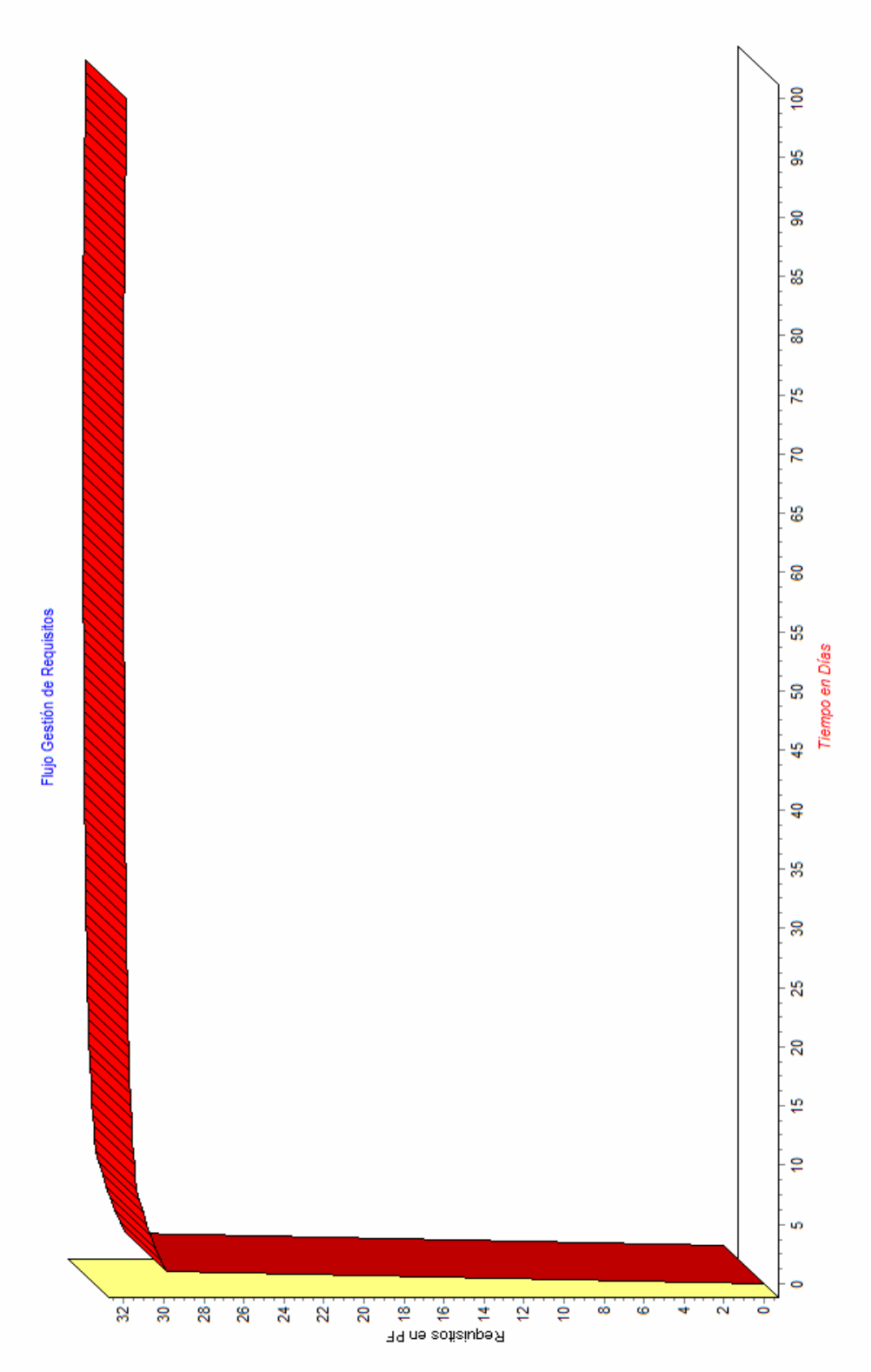


Figura IV.37. Comportamiento de la variable de flujo “Gestión de Requisitos”

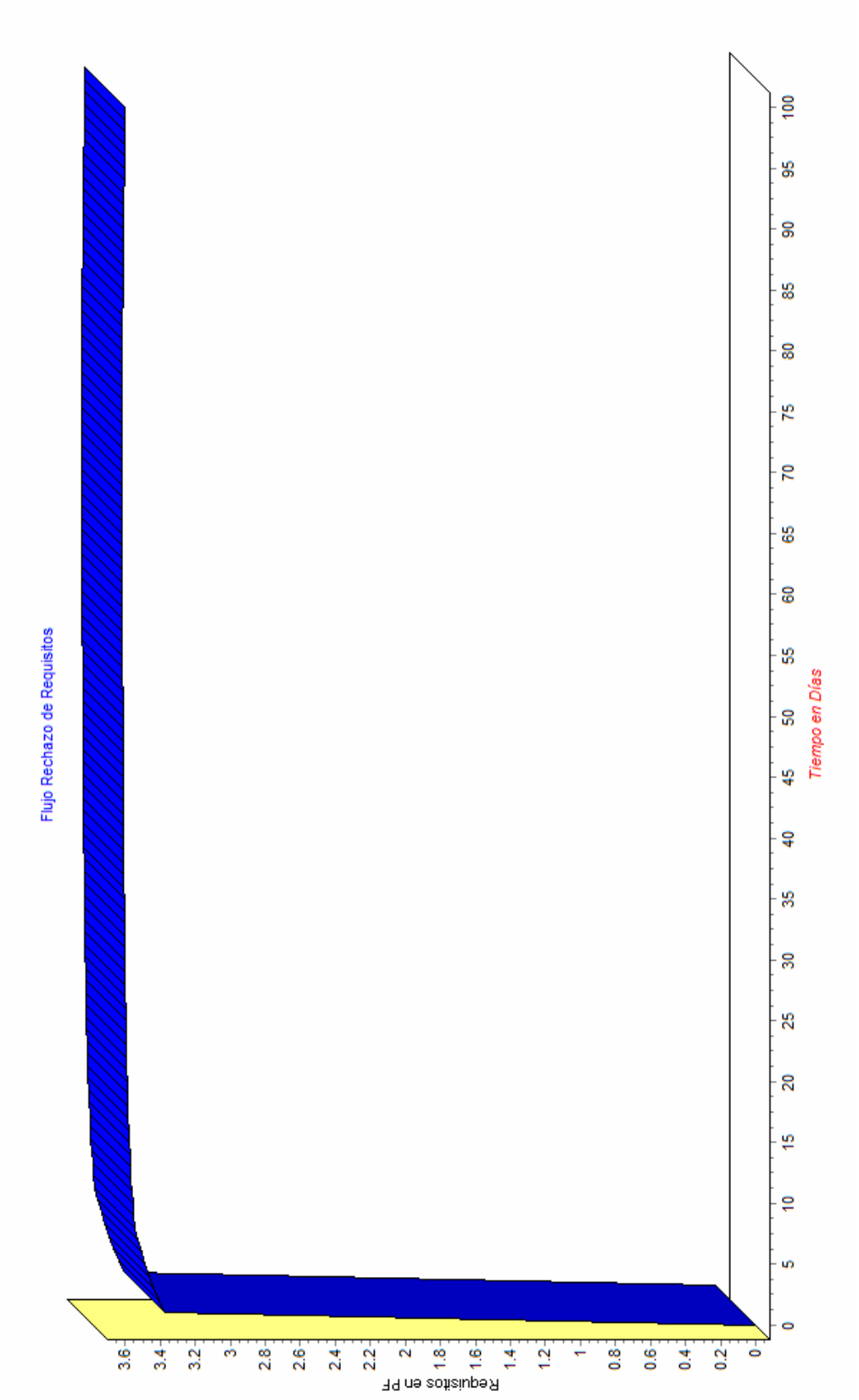


Figura IV.38. Evolución temporal de la variable de flujo “Rechazo de Requisitos”

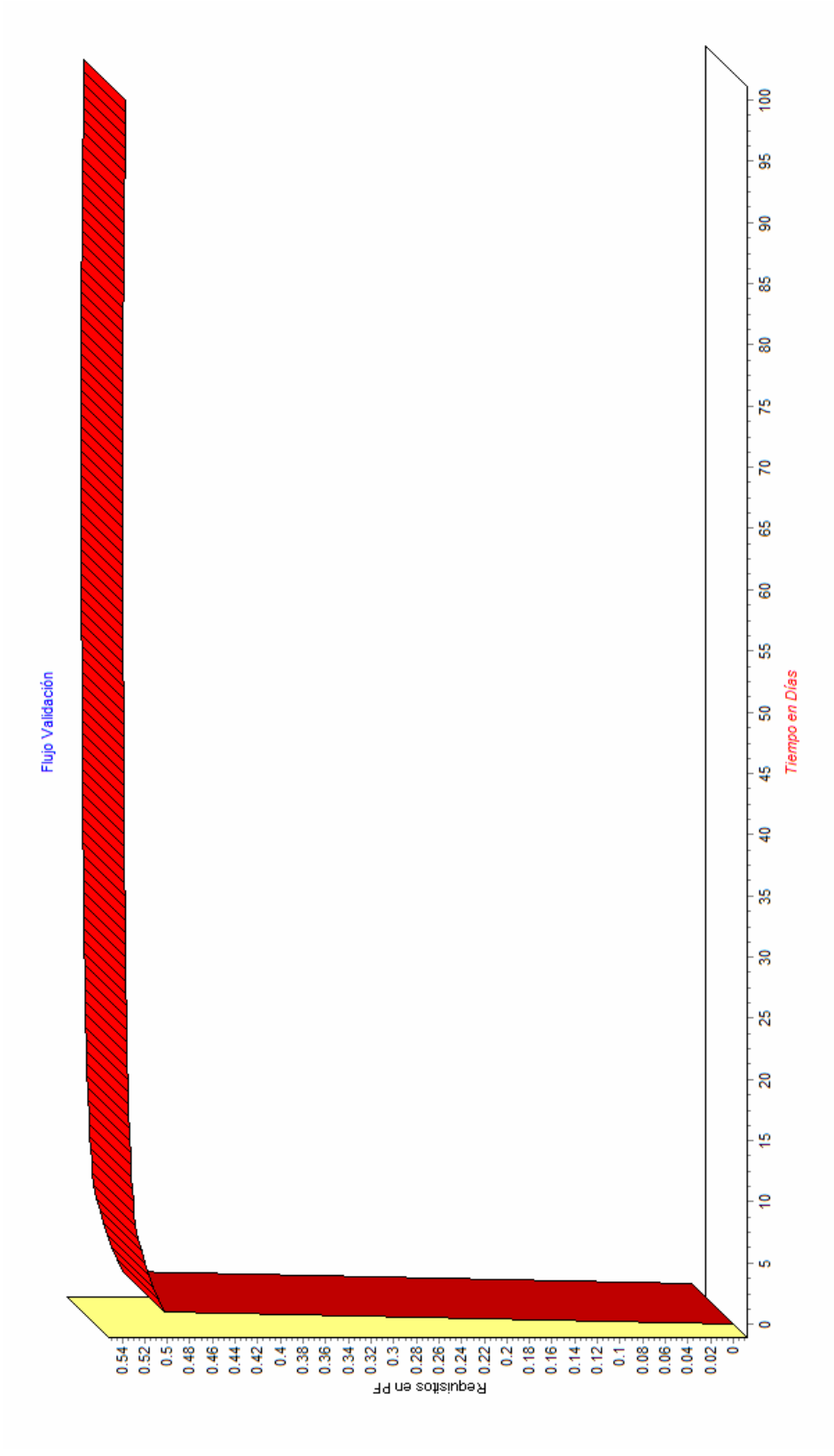


Figura IV.39. Trayectoria de la variable de flujo “Validación”

- Generación de Errores de Análisis (GeneraErrAnalis)

La generación de errores de análisis, definida por la variable "*GeneraErrAnalis*" aumenta a medida que avanza el proyecto y es un comportamiento esperado, ya que como depende de "*ReqInadecuados*", su tendencia de comportamiento es análoga (determinada por una proporción) al que sigue este nivel.

En la Figura IV.40 se puede visualizar dicho comportamiento.

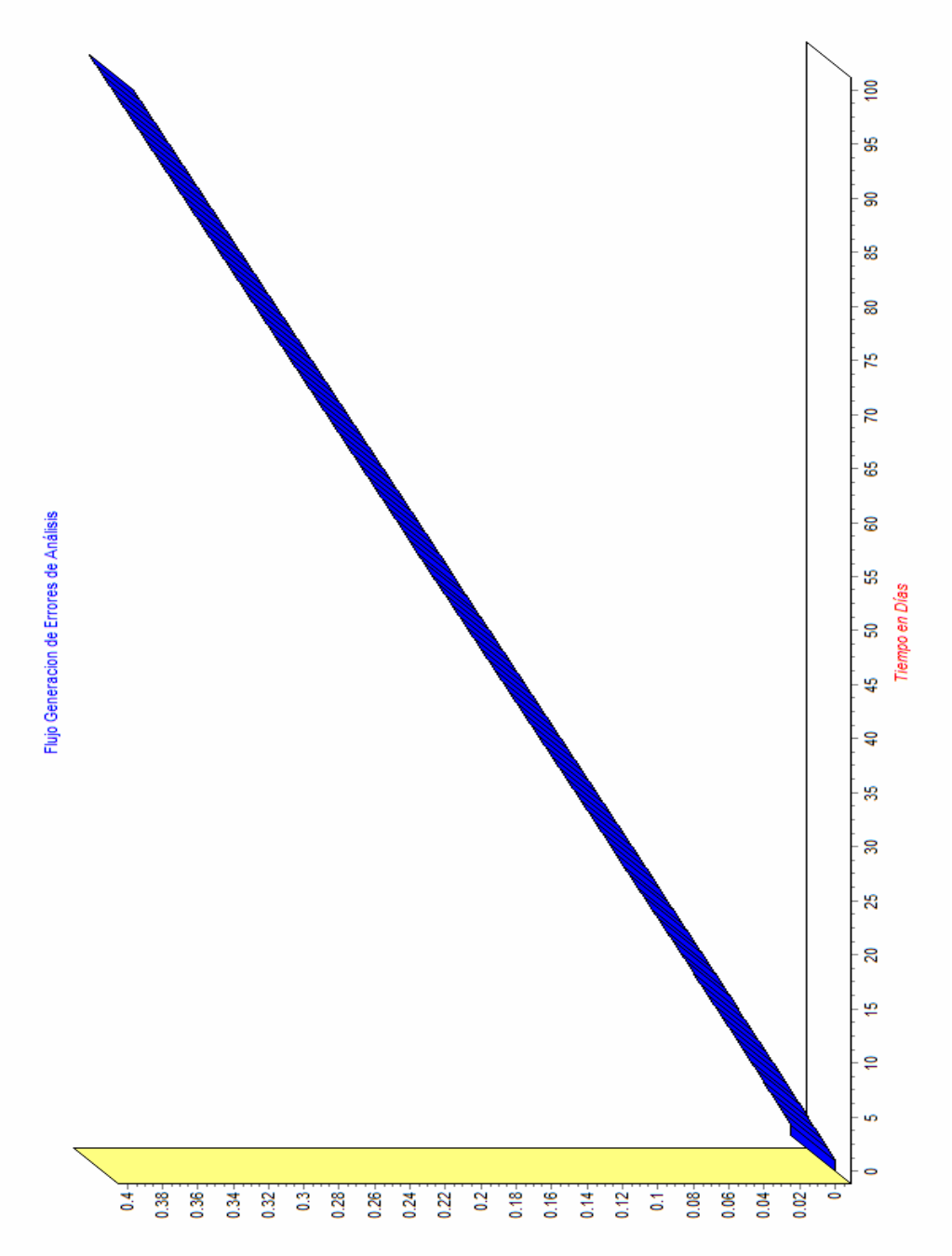


Figura IV.40. Comportamiento de la variable de flujo "Generación de Errores de Análisis"

- ***Auxiliares:***

Las variables auxiliares del modelo: *Funcionalidad Inicial, Esfuerzo de Especificación, Número de días para la etapa de análisis, Medida de la exactitud de las especificaciones,* para esta primera simulación, toman valores constantes por estar definidos en función de valores de parámetros. La única variable auxiliar que varía a lo largo de la simulación es *Productividad de especificación* y su comportamiento ya ha sido descripto al explicar el del flujo *Especificación* y el del nivel *Requisitos Especificados.*

Hasta esta instancia del desarrollo del trabajo, las variables auxiliares toman valores constantes, luego en la prueba, se podrá analizar su comportamiento con valores variables. Por el momento, son sólo proporciones.

Ya con esta única simulación (y como era previsto por ser este modelo propuesto, un modelo determinista puro) se puede observar que los resultados del calibrado, para los valores de los parámetros de entrada de la Tabla IV.2, muestran valores coherentes de salida para las variables analizadas, por lo que se puede asumir que el modelo reproduce un comportamiento del sistema real según lo esperado.

IV.1.7. *Análisis de sensibilidad*

Si este modelo es capaz de representar los diferentes estados del sistema referidos a un período de tiempo transcurrido, entonces, es posible pensar que es capaz de mostrarnos qué podría suceder en el futuro, y qué pasaría si alguno de sus parámetros fuesen cambiados en determinado momento.

El análisis de sensibilidad permite realizar una serie de modificaciones en los valores anteriores de los parámetros con el objeto de poder medir, evaluar y comparar los efectos que estos cambios tienen sobre el comportamiento del modelo.

Este análisis consiste en darles diferentes valores a los parámetros, por debajo y por encima del valor actual, viendo así el comportamiento del modelo bajo diferentes escenarios “optimistas” y “pesimistas”.

Cada uno de estos cambios puede analizarse por separado para cada una de las variables de interés, o bien considerar más de un cambio y analizar el efecto conjunto de

los mismos. Este análisis permite determinar cuales variables son más o menos sensibles a los cambios de los distintos parámetros.

El análisis de sensibilidad sólo se realizará teniendo en cuenta los parámetros "*NumEmplAnalisis*" (número de empleados para la etapa de análisis de requisitos) y "*PorcEfzoEspedif*" (Porcentaje de esfuerzo asignado a la etapa de análisis de requisitos), dado que sus valores fueron asignados según apreciaciones particulares y específicas que resultaron del estudio para el desarrollo del presente trabajo.

Los restantes parámetros, o bien son constantes y tienen valores asignados por definición según fórmulas establecidas por autores de otros estudios tomados como referencia para este trabajo, o bien, son resultados de estadísticas correspondientes a otros estudios. Sin embargo, luego, en la etapa de prueba del modelo con el caso de estudio, podrán cambiarse los valores de estos últimos parámetros e incluso definir nuevos escenarios en que cambien no sólo los parámetros sino también la definición de otras variables.

- ***Parámetro "NumEmplAnálisis"***

- En relación a "NumDíasEtapaReq"

Se ejecutó la simulación con cinco iteraciones para valores del parámetro por debajo y por arriba del valor determinado en la etapa de calibrado, 3.2, con variaciones de una unidad.

Es lógico esperar que al aumentar el número de empleados, para realizar alguna tarea, tome menor cantidad de tiempo que llevaría realizar la misma tarea pero con menor número de empleados.

Así también lo muestran los resultados de esta ejecución, como se puede apreciar en las Figuras IV.41 y IV.42.

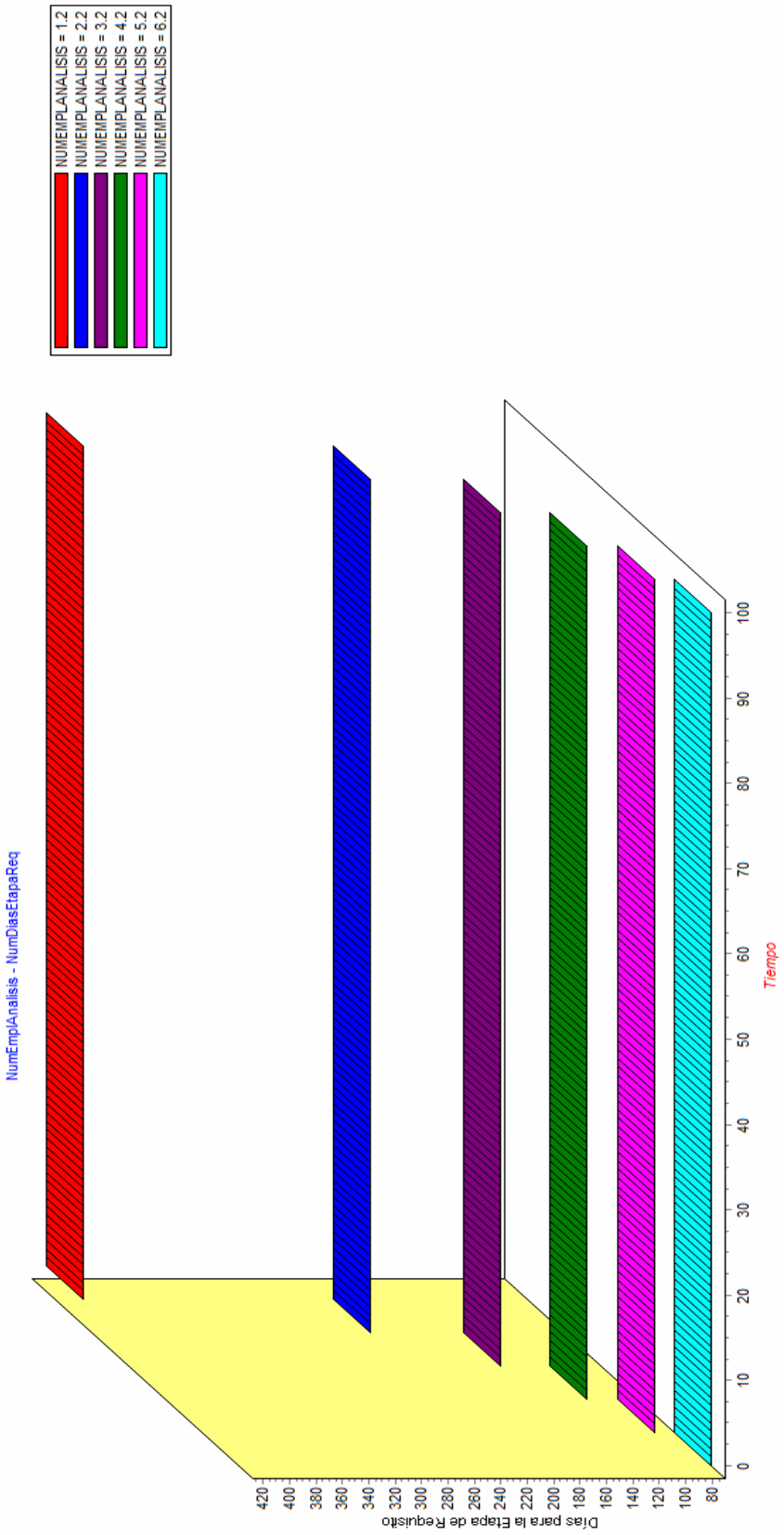


Figura IV.41. Análisis de sensibilidad de la variable “NumDiasEtapaReq” en relación a cambios en el parámetro “NumEmplAnalisis” (representación cartesiana)

Iterac.	X:T	NUMEMPLANALISIS = 1.2	NUMEMPLANALISIS = 2.2	NUMEMPLANALISIS = 3.2	NUMEMPLANALISIS = 4.2	NUMEMPLANALISIS = 5.2	NUMEMPLANALISIS = 6.2
1	0	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
2	1	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
3	2	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
4	3	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
5	4	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
6	5	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
7	6	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
8	7	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
9	8	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355
10	9	417	227.45454545455	156.375	119.14285714286	96.230769230769	80.709677419355

Figura IV.42. Análisis de sensibilidad de la variable “NumDiasEtapaReq” en relación a cambios en el parámetro “NumEmplAnalisis” (representación tabular)

- En relación a "MedidaExacEspec"

En relación a la medida de exactitud de la especificación, y con idénticas características para la simulación, el comportamiento resultante aunque con una variación más homogénea, que se puede visualizar en las Figuras IV.43 y IV.44, es análogo a la ejecución anterior por estar esta variable en función de "*NumDíasEtapaReq*". Cuanto mayor sea el número de empleados para la etapa de análisis, menor será la distribución del trabajo a realizar, en el tiempo disponible.

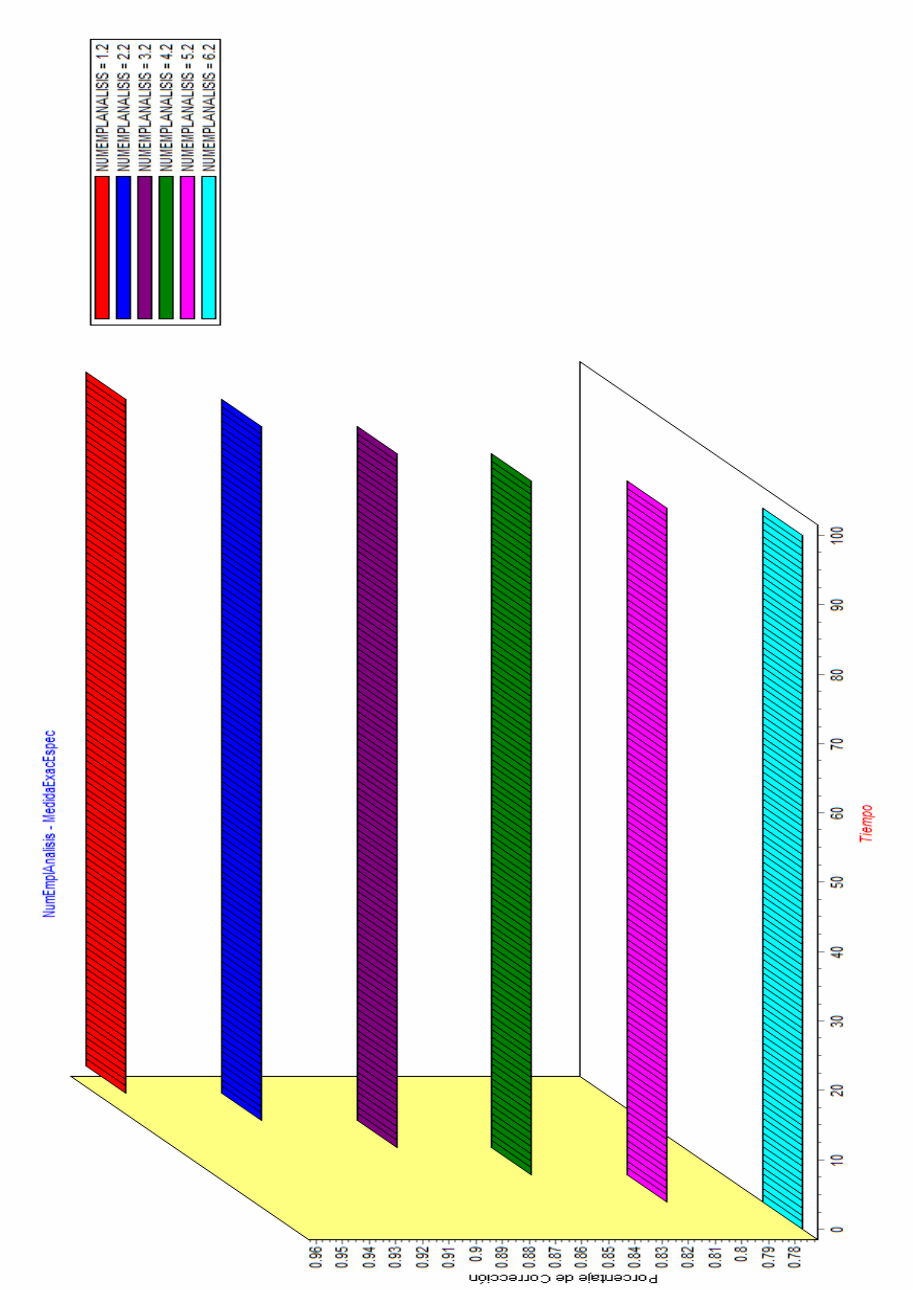


Figura IV.43. Análisis de sensibilidad de la variable "MedidaExacEspec" en relación a cambios en el parámetro "NumEmplAnalisis" (reptac. cartesiana)

Iterac.	X:T	NUMEMPLANALISIS = 1.2	NUMEMPLANALISIS = 2.2	NUMEMPLANALISIS = 3.2	NUMEMPLANALISIS = 4.2	NUMEMPLANALISIS = 5.2	NUMEMPLANALISIS = 6.2
1	0	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
2	1	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
3	2	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
4	3	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
5	4	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
6	5	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
7	6	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
8	7	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
9	8	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989
10	9	0.956909472422062	0.921000699440447	0.885091926458833	0.849183153477218	0.813274380496603	0.777365607513989

Figura IV.44. Análisis de sensibilidad de la variable “MedidaExacEspec” en relación a cambios en el parámetro “NumEmplAnalisis” (representación tabular)

- ***Parámetro "PorcEfzoEspecif"***

- En relación a "ProducEspecific"

Para valores por debajo y por encima de 9% que es el valor determinado en el calibrado para "*PorcEfzoEspecif*", se observa la incidencia en forma indirecta a través de la variable "*EfzoEspecific*", sobre la variable "*ProducEspecific*" que determina su cambio en el mismo sentido. Este es un comportamiento esperado ya que a mayor esfuerzo diario, mayor será la productividad y se puede observar en las distintas representaciones de las Figuras IV.45 y IV.46.

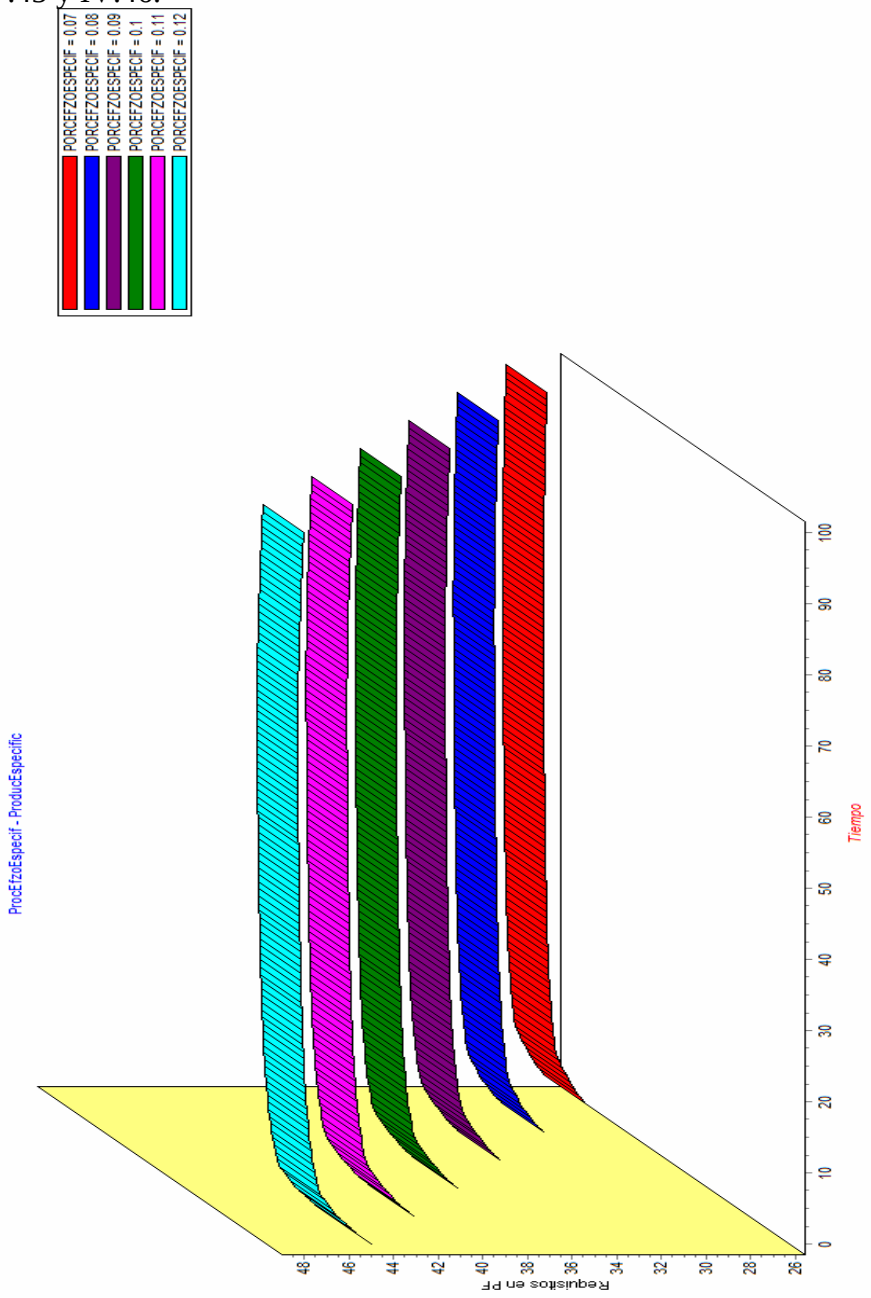


Figura IV.45. Análisis de sensibilidad de la variable "ProducEspecific" en relación a cambios en el parámetro "PorcEfzoEspecif" (representación cartesiana)

Iterac.	X-T	PORCEFZOESPECIF = 0.07	PORCEFZOESPECIF = 0.08	PORCEFZOESPECIF = 0.09	PORCEFZOESPECIF = 0.1	PORCEFZOESPECIF = 0.11	PORCEFZOESPECIF = 0.12
1	0	26.25899635514	30.0102818691588	33.7615671028037	37.5128523364486	41.2641375700934	45.0154228037383
2	1	26.5094786957893	30.2965470809021	34.0836154660149	37.8706838511276	41.6577522362404	45.4448206213532
3	2	26.7357471360797	30.5551395840911	34.3745320321025	38.1939244801139	42.0133169281252	45.8327093761366
4	3	26.9399947813058	30.7856564643495	34.6371361473932	38.4857068304369	42.3342775134806	46.1828481965243
5	4	27.1243300649008	30.9992343598866	34.8741386548724	38.7490429498583	42.6239472448441	46.4988515398299
6	5	27.2906689021812	31.1893358882071	35.0880028742329	38.9866698602588	42.8853368462847	46.7840038323106
7	6	27.4407512394 37	31.3608585593566	35.2809658792762	39.2010731991957	43.1211805191153	47.0412878390349
8	7	27.5761562327 72	31.515607123168	35.4550580 13564	39.3945089 0396	43.3339597943 56	47.2734106847 52
9	8	27.6396367405982	31.5881562749694	35.5366758093406	39.4851953437118	43.4337148790829	47.3822344124541
10	9	27.6812780474656	31.6357463399607	35.5902146324558	39.5446829249509	43.4991512174 46	47.4536195099411

Figura IV.46. Análisis de sensibilidad de la variable “ProducEspecific” en relación a cambios en el parámetro “PorcEfzoEspecif”(representación cartesiana)

Si bien el modelo es determinista puro y sólo con la primera simulación bastaba para demostrar que el conjunto de parámetros están ajustados a la realidad, cambiando los parámetros a valores optimistas y pesimistas el modelo sigue reproduciendo un comportamiento semejante al esperado con variaciones proporcionales. No se registra sensibilidad extrema en relación a los cambios de parámetros.

IV.1.8. *Evaluación del modelo (Contrastado)*

En las etapas de Calibrado y Análisis de Sensibilidad, el modelo se ha ido ejecutando, variando los parámetros de entrada para analizar los resultados arrojados, sólo como tendencia general, comparados con datos encontrados en la bibliografía de referencia.

Ahora, con el modelo ya calibrado, la evaluación implica un contraste, pero con datos reales, para determinar si el modelo arroja datos coherentes; es decir, hay que comparar los datos reales obtenidos del sistema, con los datos obtenidos de la simulación para asegurarse de que el modelo arroje resultados confiables. Si en el contraste para variables principales, los resultados no se encuentran dentro de un margen de error aceptable se debe revisar el modelo.

Aunque se puede decir que el modelo propuesto es coherente según los resultados de las dos etapas anteriores, la validación del modelo completo se alcanzará en el Capítulo siguiente con la prueba general del modelo con el caso de estudio cuando ya se disponga de datos reales que permitan la comparación.

IV.1.9. *Utilización del modelo*

Esta etapa ya no es de construcción del modelo, sino que justifica su creación: *el modelo debe ser utilizable*. La primera utilización de un modelo de Dinámica de Sistemas, es una aproximación al conocimiento del sistema que representa y a partir de esto se lo usa para simular situaciones supuestas del sistema real [20]. Simular es la generación de valores de las variables endógenas (salidas) a partir del modelo, con diversas hipótesis alternativas de conjuntos coherentes de valores de las variables exógenas y parámetros (entradas). Esto implica conocer la *estructura del modelo* (conjunto de relaciones) y determinar las *condiciones iniciales* (valores fijos iniciales de las variables de nivel, flujos

y auxiliares del modelo) y *escenarios de simulación* (valores coherentes de variables exógenas y parámetros).

Como en la etapa anterior, determinar las condiciones iniciales y los escenarios para poder simular el modelo y probar si es o no utilizable, requiere de los datos reales que nos proporcionará el caso de estudio.

Por ello, una descripción en detalle de esta etapa se indicará en el próximo Capítulo, cuando el modelo se simule bajo ciertas condiciones iniciales y diferentes escenarios establecidos a partir de los datos proporcionados por el Proyecto DE-A seleccionado como caso de estudio en el presente trabajo, para probar su hipótesis.

IV.2. *Interfase del Modelo Propuesto con el Modelo Base*

El modelo descrito hasta el momento se corresponde con una abstracción particular de la etapa de análisis del proceso de desarrollo de software.

Es necesario también lograr un visión de acoplamiento / integración de este modelo con el modelo base y poder cuantificar el impacto de la dinámica de la etapa de análisis, en el resto del proceso de desarrollo, en relación a costos, tiempos, calidad.

Si reducimos la perspectiva alcanzada desde el diagrama de Forrester del modelo propuesto (nivel de mayor formalización) a un símbolo de sector, obtenemos un esquema de bloques, con el que se puede destacar la vinculación con diferentes sectores y sub-sectores del modelo base (ver Figura IV.47):

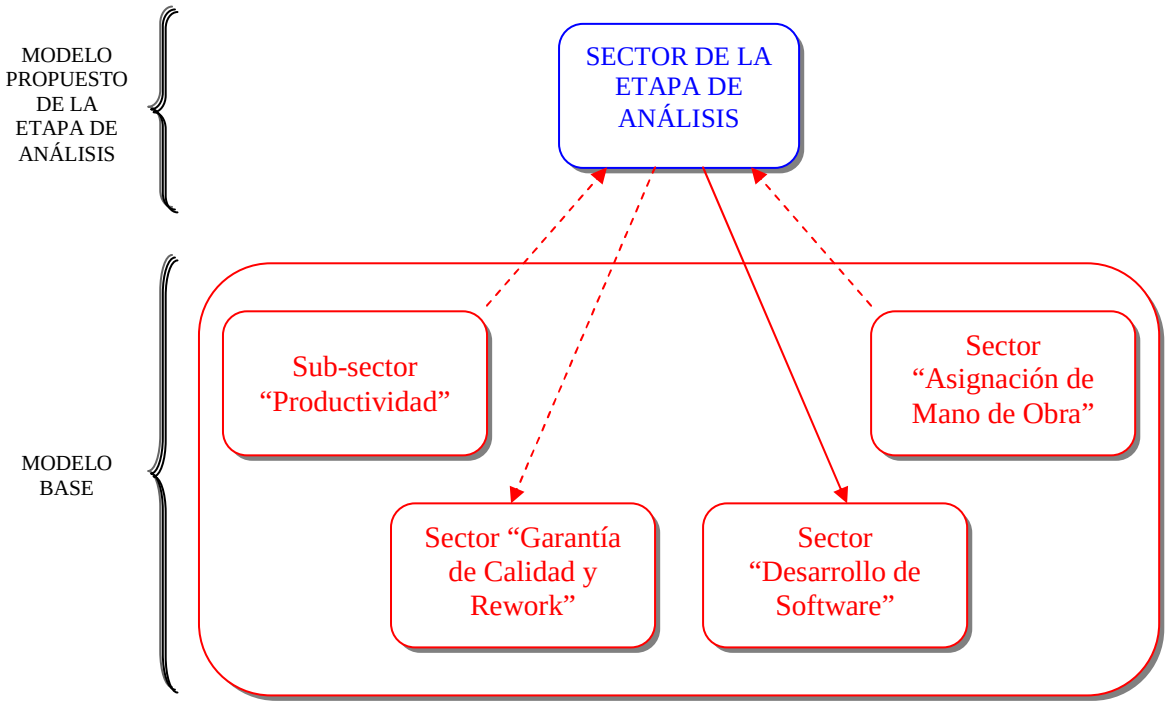


Figura IV.47. Relación del Modelo Propuesto con Modelo Base usando Diagrama de Bloques

La figura precedente permite visualizar como el sector del modelo propuesto se encuentra conectado a los distintos sectores del modelo base, no sólo a través de flujos de información sino también a través de flujos de material. Las variables que hacen de interfase entre los modelos están resaltadas en color rojo en la Figura IV.15.

Aunque el diagrama de bloques muestra conexiones explicitas, también se pueden analizar relaciones que se dan implícitamente derivadas de las primeras.

A continuación se describen las relaciones del modelo de la etapa de análisis con los diferentes sectores y subsectores del modelo original. Luego se describirán las relaciones con el resto de los subsistemas aunque no tengan relación explícita con el modelo propuesto.

IV.2.1. *Relaciones Explícitas*

1).- Relación del modelo de análisis con los subsistemas de *gestión de recursos humanos y asignación de mano de obra.*

En relación a estos dos subsistemas interesa conocer qué caracteriza el flujo de personal afectado a la actividad de análisis y cómo este flujo impacta en el resto del proceso de desarrollo.

En este punto es importante analizar los resultados de la manipulación de las variables como "*Mano de obra diaria promedio por empleado*", "*Mano de obra recientemente contratada*", "*Mano de obra experimentada*" que, entre otras, influyen en la productividad y ésta a su vez en los resultados alcanzados en la producción de software considerando que ésta, ahora, se extiende desde la definición de requisitos.

En el modelo de análisis se define la variable que representa la mano de obra asignada a esta etapa, "*Esfuerzo de Especificación*" y que corresponde a un porcentaje del "*Esfuerzo Total*" asignado al proyecto y juntamente con la variable "*Productividad de Especificación*", que se asume idéntica a la "*Productividad de Desarrollo*", controlan la producción de las especificaciones.

En cuanto a esta relación, controlando cambios en el valor del porcentaje de mano de obra diaria asignado a la actividad de especificación ("*Esfuerzo de Especificación*"), sería posible estudiar cómo la cantidad de esfuerzo afecta a los resultados de costo, tiempo y calidad.

2).- Relación del modelo de análisis con el subsistema de *producción de software* (incluye el sector de desarrollo de software y subsector de productividad de desarrollo).

Para los autores del modelo base, el proceso de producción consiste en el diseño, codificación y prueba del producto software. Con la incorporación a este modelo de la dinámica de la etapa de análisis, el sector de desarrollo del subsistema de producción consistirá ahora del análisis, diseño y codificación. Por lo tanto el progreso en la producción se definirá también por el número de tareas realizadas a nivel de definición de requisitos.

El vínculo específico entre el modelo de análisis y el modelo base se establece a través de la variable de flujo "*Tasa de desarrollo de software*". Este vínculo une el nivel de "*Requisitos Validados*" del modelo de análisis con el nivel "*Tareas desarrolladas*" del modelo tomado de referencia.

Esto puede interpretarse en el sentido que, a partir de los requisitos validados se inicia el desarrollo del diseño y posteriormente la codificación.

Esta interfase es fundamental por el hecho de acoplar la actividad de producción en sus diferentes fases y al mismo tiempo por regular uno de los cinco números en los que se basa la gestión de proyectos, "la medida de la cantidad producida", ya que la variable "*Tasa de desarrollo de software*" es el flujo que alimenta el nivel "*Tareas desarrolladas*".

Nota: Esta interfase implica adecuar la unidad de medida de tamaño de puntos de función (utilizada en el modelo de análisis) a líneas de código (utilizada en el modelo base). Esto conversión se realiza con la ayuda de un parámetro según equivalencia utilizada para la definición de la variable de flujo "*Elicitación*".

3).- Relación del modelo de análisis con el subsistema de *garantía de calidad (QA) y rework*

Es reconocida la incapacidad humana para realizar con perfección el desarrollo del software por lo que es incorporado a éste la actividad de garantía de calidad (QA).

Como expresan Abdel-Hamid y Madnick, la QA de software incluye dos metodologías distintivas y complementarias. La primera es el diseño de un coherente, completo, no ambiguo y no conflictivo conjunto de requisitos. La segunda es la revisión y prueba del producto.

La primera metodología es fundamental ya que errores pueden ocurrir al principio del proceso donde los objetivos del sistema software pueden ser especificados errónea o incompletamente.

Sin embargo esta metodología no es tratada en el modelo base, ya que los autores asumen que los requisitos están perfectamente especificados, y que no sufrirán cambios significativos subsecuentes, con los cuales se procede al diseño y código donde tales requisitos ya son mecanizados.

En relación a esta consideración y sabiendo que cambios y omisiones en la definición de requisitos son comunes y causales de desbordes en tiempo y costo del proyecto, generados por errores no detectados y no corregidos a tiempo, se desarrolla el modelo de análisis que contempla el proceso iterativo de *gestión de requisitos* con lo que se pretende realizar las adecuaciones necesarias en la definición de requisitos para asegurar la satisfacción del usuario y a la vez, realizar sucesivas correcciones en etapas tempranas para evitar que la propagación hacia fases posteriores de la construcción provoque que los errores sean más graves y difíciles de corregir/reparar, reduciendo así esfuerzo, tiempo, y por lo tanto el costo.

Además, no sólo se consideran los cambios y omisiones de requisitos sino también la introducción/inyección de nuevos requisitos y con esto la generación de errores en las

actividades de análisis y especificación. Como lo manifiestan los autores del modelo de referencia, las tasas de generación para los distintos tipos de errores, son mayores en las etapas iniciales. Por lo tanto considerar los errores generados en las actividades de análisis y especificación es crucial para mejorar la calidad del desarrollo.

Esta revisión intensa desde los inicios del proceso de desarrollo, se supone reducirá errores en análisis, y por consiguiente, se lograrían diseños y códigos más puros / limpios.

Así como se describieron las relaciones explícitas entre el modelo propuesto y el modelo base, a continuación se describen aquellas relaciones, que si bien no tienen una representación visible en las formalizaciones mostradas a lo largo del presente Capítulo, muestran una tendencia probable. Se espera que en la etapa de simulación del modelo integral puedan confirmarse resultados positivos a estas tendencias analizadas en las siguientes descripciones.

IV.2.2. *Relaciones Implícitas*

1).- Relación del modelo de análisis con el subsistema de *prueba.*

Se sabe que los errores no detectados por QA y las correcciones malas que resultan del rework no permanecen inactivos hasta ser detectados y corregidos en la etapa de prueba, sino que tienen una existencia activa, produciendo más y más errores en el sistema.

Por esta misma razón, detectar y corregir un error de diseño durante la fase de diseño consume un décimo del esfuerzo que sería necesario para detectarlo y corregirlo durante la fase de prueba porque también implicaría hacer correcciones en el inventario de especificaciones, código, manuales, etc.

De aquí que poder detectar y corregir errores durante la etapa de análisis llevaría a un menor consumo de esfuerzo, tiempo y costo. El ciclo de gestión de requisitos pretende lograr detecciones tempranas de errores y, por consiguiente, análisis y especificaciones más puros (menos errores), como se explica en el apartado 3).- de Relaciones Explícitas, y por ende reduciría el esfuerzo, tiempo y costo insumidos en la etapa de prueba. En este sentido se habla de una reducción en la escalación del costo, ya que al reducir el número de errores (densidad de error) que pasa de una etapa a la otra, se reduce también la regeneración de errores activos.

2).- Relación del modelo de análisis con el subsistema de *control*.

Al resultar especificaciones más puras por la implementación del ciclo "gestión de requisitos" en el modelo de análisis, es de esperar que las mediciones y evaluaciones del estado/progreso del proyecto en términos de esfuerzos percibidos restantes (para desarrollar nuevas tareas, detectar errores, rework errores detectados y probar tareas) sean menores que los que se obtendrían si no se hubiera incorporado este ciclo mencionado. Esto porque al disminuir la cantidad de errores desde el análisis, se disminuye también el esfuerzo dedicado al rework y prueba y a la vez será menor el esfuerzo que se tenga que desviar para ello, desde las actividades de desarrollo de tareas, lo que mejora los valores de productividad favoreciendo a los valores globales de tiempo, costo y calidad del proyecto.

3).- Relación del modelo de análisis con el subsistema de *planificación.*

En cuanto a las mediciones y evaluaciones del estado/progreso del proyecto en términos de tiempos percibidos restantes, al igual que en la relación anterior con respecto a los esfuerzos, es posible a partir de reducciones en la cantidad de errores desde el análisis (con la implementación del ciclo "gestión de requisitos"), disminuir los tiempos insumidos en actividades de rework, prueba y desarrollo de nuevas tareas. O bien, en casos que no puedan disminuirse, al menos no llevar a excesos que impliquen decisiones de la gestión de pagar cualquier precio por evitar superar el máximo tolerable de fecha de entrega, casos en los que la decisión es contratar más empleados, con sus efectos negativos ya indicados y que consecuentemente llevan a un mayor costo.

El razonamiento de las relaciones tanto explícitas como implícitas, descriptas precedentemente, guía el proceso de prueba del modelo completo y que se desarrolla en el siguiente Capítulo.

CAPITULO V

.....

PRUEBA DEL MODELO

Presentación

Teniendo en cuenta que el estudio y construcción de un modelo de simulación implica la ejecución de las siguientes tres principales fases: *definición del problema*, la *planificación de la simulación* y la *ejecución u operación de la simulación*; el presente capítulo es la culminación de la tercera fase.

En esta fase de ejecución de la simulación, el modelo se simula para descubrir los parámetros de salida y relacionarlos básicamente con las estimaciones de tiempo y esfuerzo. El modelo se verifica y valida para asegurar su credibilidad, relevancia y utilidad. Esta fase comienza con la traducción del conjunto de las ecuaciones matemáticas a la herramienta de simulación, lo cual se realizó en el Capitulo IV y se completa con los ajustes necesarios indicados desde la prueba del modelo con el caso de estudio, que se encara en el presente capítulo, para asegurar que el modelo trabaja apropiadamente.

Resumiendo, en este capítulo se presentan los resultados de simulación de un caso de estudio que conduce la prueba del modelo. El objetivo es examinar la habilidad del modelo para reproducir los patrones dinámicos de un proyecto de software ya terminado.

V.1. *Cuestiones Generales que guían la Operación de la Simulación*

El modelo final se ejecuta en una serie de simulaciones para responder a los siguientes interrogantes:

- ¿Es posible introducir **mejoras en el proceso de gestión de proyectos software** mediante la incorporación de modelos dinámicos del proceso de Ingeniería de Requisitos a los modelos de soporte a la gestión?
- ¿La formalización de un modelo dinámico de gestión, centrado en la mejora de la etapa de análisis, puede incrementar la predictibilidad de **costos** y **tiempos** del proyecto software?
- ¿La formalización de un modelo dinámico de gestión, centrado en la mejora de la etapa de análisis, puede incrementar la **calidad** del producto software?
- ¿La incorporación de un modelo del proceso de Ingeniería de Requisitos a los modelos de soporte de la gestión, puede aumentar la **productividad** en el proceso de desarrollo de software?

Los resultados de las ejecuciones de las simulaciones se presentan en gráficas y tablas, pero dado que se estudia un modelo de Dinámica de Sistemas, los resultados deben interpretarse cualitativamente, es decir, la importancia de éstos está en las tendencias y no en los porcentajes exactamente.

V.2. *El Caso de Estudio: Proyecto DE-A [1]*

El proyecto DE-A (Dynamics Explorer-Attitude) fue un proyecto dirigido por el departamento de sistemas de la NASA, organización comprometida con el desarrollo de software para dar soporte al control y determinación de las acciones de naves espaciales.

Los requerimientos para el Proyecto DE-A fueron diseñar, implementar y probar un software que pueda procesar datos telemétricos y determinar las posiciones definitivas así como el soporte de control y determinación de posiciones en tiempo real para el satélite DE-A. Este satélite fue diseñado para estudiar la atmósfera, ionosfera y magnetosfera de la Tierra.

Los requisitos que se habían definido en el marco empírico para la selección del caso de estudio fueron que estén disponibles cómo mínimos los datos de: Tamaño, Duración y Mano de Obra. En cuanto al tamaño se eligió el de mediana envergadura.

Luego, este proyecto fue seleccionado como caso de estudio por satisfacer los siguientes criterios: 1). Mediana envergadura, es decir tamaño medio (entre 16-64 KSDI), 2). Proyecto "típico", es decir desarrollado en un entorno familiar, 3). El tamaño del proyecto fue de 24.400 instrucciones fuente, 4). Fue terminado en 19 meses calendario y 5). Consumió 2.222 hombre/días de esfuerzo.

V.2.1. *Parametrización del Modelo para el Caso de Estudio*

Para simular esta situación de proyecto particular, es necesario establecer los siguientes cuatro conjuntos de parámetros:

- ***Gestión de Recursos Humanos***

  a.- Mano de obra diaria promedio por empleado

  b.- Retraso de contratación

  c.- Tiempo de empleo promedio

d.- Entrenadores por contratado reciente

e.- Retraso promedio de asimilación

- ***Entorno de Desarrollo de Software***

  a.- Productividad potencial promedio

  b.- Tasa de error

  c.- Distribución de esfuerzo entre desarrollo y prueba

  d.- Asignación de esfuerzo para QA

- ***Entorno de Planificación***

  a.- Máxima fecha de compleción tolerable

- ***Estimaciones Iniciales del Proyecto***

  a.- Tamaño del proyecto

  b.- Esfuerzo gastado en el proyecto

  c.- Duración del proyecto

Los parámetros para el Modelo DE-A se resumen el la Tabla V.1. Ellos están referenciados por sus nombres usados en el modelo equivalente en Evolution.

Nombre del Parámetro	Unidad	Valor
MODiariaPromXEm	Dimensional	0,5
RetrasoContrato	Días	30
TpoPromDeEmpleo	Días	1000
EntrenXContrRec	Dimensional	0,25
RetrasoPromAsim	Días	20
InsFteEntXTarea	LOC/Tarea	40
ErrNormComXIFEn	Errores/Tarea	24/22,9/20,75/15,25/13,1/12
PorEzoAsuNecDes	Dimensional	0,85
FraccPlanMOQA	%	0,325/0,29/0,275/0,255/0,25/0,275/0,325/0,375/0,4/0,4/0
MaxFechaCompTol	Dimensional	1,16

Tabla V.1. Valores de los parámetros del Modelo DE-A

Nombre del Parámetro	Unidad	Valor
TamRealTrabDSI	LOC	16000 (35% de subestimación)
TamTotTrabEzo	Hombres/Día	1111 (50% de subestimación)
FechaCompProgra	Días	320

Tabla V.1. Valores de los parámetros del Modelo DE-A (continuación)

Estos parámetros (valores estimados) se suman a los ya indicados en el capítulo anterior y que constituyen los parámetros del modelo propuesto de la etapa de análisis y que muestran en la Tabla V.2:

Nombre del Parámetro	Unidad	Valor
TasaGenerNvsReq	%	2
TasaRechazoReq	%	10
CteConversionPF	Ctte.	128
PorcEfzoEspecif	%	20
CtteMedExac1	Ctte.	1
CtteMedExac2	Ctte.	0,02
DiasTrabajoMes	Días	20
NumEmplAnalisis	Empleado	3,2

Tabla V.2. Valores de los parámetros del modelo propuesto de la Etapa de Análisis

V.3. *El Comportamiento del Proyecto Simulado y Real*

Luego que el modelo ha sido parametrizado, se ejecuta para simular el proyecto DE-A. En esta instancia se analizan las salidas que proporciona el modelo y se las compara con el comportamiento real del proyecto.

En relación a la hipótesis planteada para este trabajo, interesa examinar la dinámica del comportamiento para las siguientes variables del proyecto:

V.3.1. *En Relación al Tiempo*

- **Fecha de Compleción Programada** (*FechaCompProgra*)

Las Figuras V.1 y V.2 muestran, en representación cartesiana y tabular respectivamente, la trayectoria de la variable fecha de compleción estimada, medida en términos del número de días transcurridos hasta la finalización del proyecto.

El modelo capta con precisión la tendencia de la política de la gestión a no ajustar la fecha de finalización del proyecto durante la mayor parte del desarrollo.

Al ejecutar el modelo para 380 iteraciones, que es el tiempo real de duración del proyecto (19 meses de 20 días de trabajo cada uno), se puede observar como resultado de la simulación del modelo propuesto un valor mayor sólo por pocas unidades: 382,32 días.

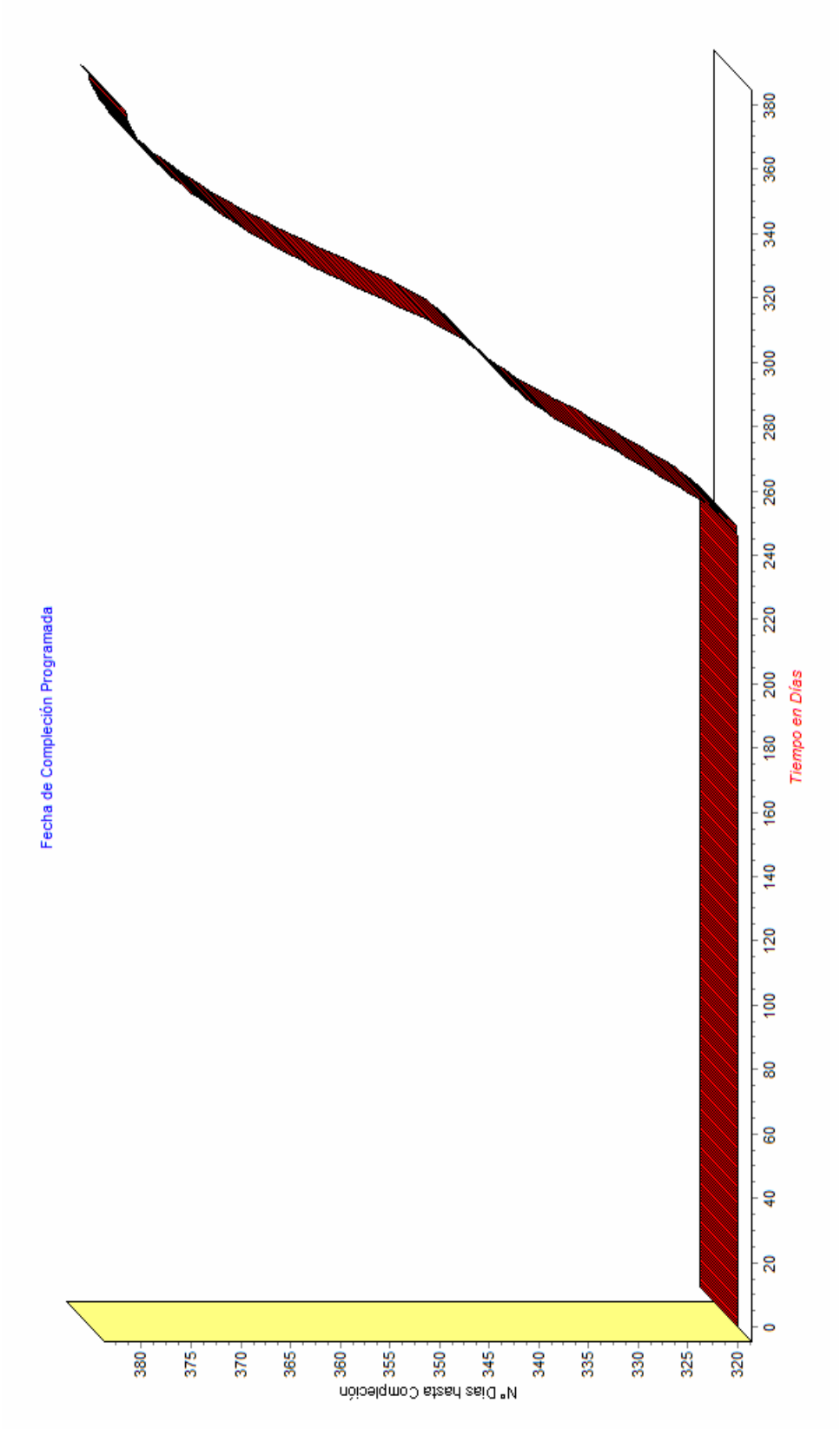


Figura V.1. Comportamiento de la variable "Fecha de Compleción Tolerable" (representación cartesiana)

Iterac.	X:T	FECHACOMPPROGRA
372	371	380.671978541592
373	372	380.843579711529
374	373	381.002639958693
375	374	381.149506030991
376	375	381.284428886603
377	376	381.408664646996
378	377	381.523065655099
379	378	381.675236572884
380	379	381.937215021411
381	380	382.316188273279

Figura V.2. Resultados de las ultimas 10 iteraciones del modelo propuesto

Sin embargo, la ejecución del modelo base muestra como resultado de la duración del proyecto 387,5 días [1]. De la comparación de estos valores de salida, podemos afirmar que el resultado producido por el modelo propuesto es más próximo al real, que el proporcionado por el modelo base. Más aún, y como se intentaba probar, al incorporar al modelo base la dinámica de la etapa de análisis se esperaba reducir los tiempos de ejecución del proyecto ya que trabajar al principio del proyecto con nuevos requisitos, cambios/gestión de requisitos, también al descartar requisitos obsoletos o redundantes para evitar trabajo innecesario, se lograría un documentos base de especificación de requisitos, para la subsiguientes etapas de diseño, codificación y prueba, más claro y limpio que evita consumo de tiempo extra.

V.3.2. *En Relación al Esfuerzo/Costo:*

- **Consumo de Esfuerzo** (*TamTotTrabEzo*)

A diferencia de lo que ocurre con la duración del proyecto, la Figura V.3 que se muestra a continuación, indica que los ajustes en el gasto estimado del total de mano de obra, medido en términos de esfuerzo, se realizan más tempranamente.

Estos ajustes se activan por el descubrimiento de tareas de trabajo no descubiertas.

Este es un comportamiento esperado ya que, como se expuso al describir el modelo base, es política de la gestión que mientras no haya presión de tiempo se mantenga la estabilidad de la mano de obra, pero que sí tendrá total disponibilidad a ajustar el tamaño del esfuerzo, a cualquier nivel necesario, para cumplir con la fecha de compleción programada del proyecto.

El resultado de la simulación del modelo propuesto indica un consumo total (al finalizar el proyecto), como se puede visualizar en la Figura V.4, de 2129,73 hombres/días de esfuerzo. Por su parte la misma ejecución con el modelo base, proporciona como salida 2092 hombres/días. Ambos son levemente menores que el valor real registrado para el proyecto (2222 hombres/días) pero el resultado de la ejecución del modelo propuesto es más cercano al valor real. Además la última salida de la es mayor que la del modelo base, lo cual es esperado ya que al incluir la etapa de análisis, aumenta el número de tareas a realizarse que antes no estaban contempladas en el modelo y que requerirán algo más del esfuerzo que se consumió para las consideraciones del modelo base.

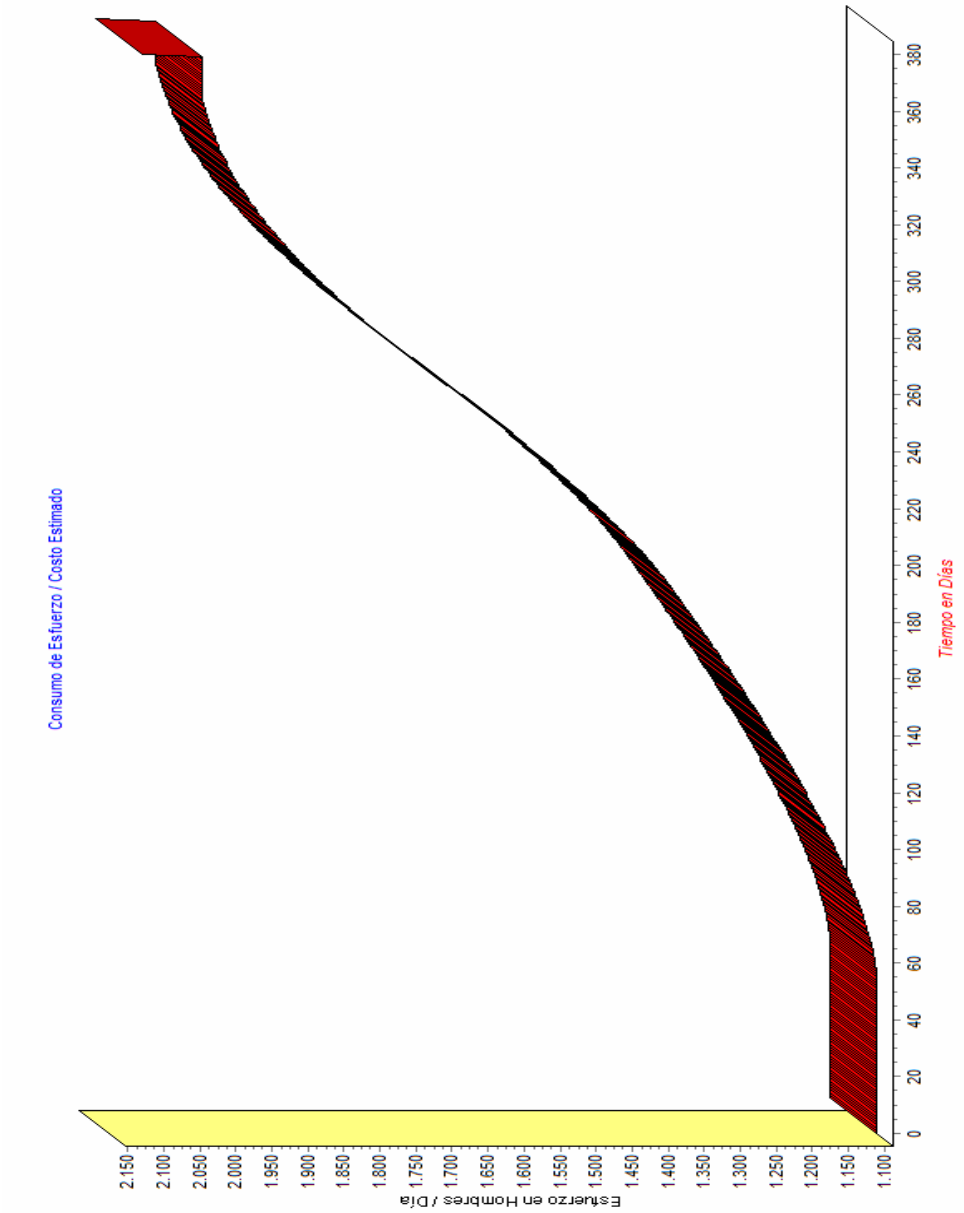


Figura V.3. Trayectoria de la variable "Tamaño Total del Trabajo en Esfuerzo" (representación cartesiana)

Iterac.	X:T	TAMTOTTRABEZO
372	371	2047.41725805377
373	372	2047.41725805377
374	373	2047.41725805377
375	374	2047.41725805377
376	375	2047.41725805378
377	376	2047.41725805378
378	377	2047.41725805378
379	378	2047.41725805378
380	379	2047.41725805379
381	380	2129.72948595191

Figura V.4. Resultados de las ultimas 10 iteraciones del modelo propuesto

Los comportamientos de las dos variables analizadas hasta aquí (Tiempo y Esfuerzo), replican correctamente las decisiones de la gestión de ajustar el plazo, el esfuerzo, o una combinación de ambos, acciones que continuarán basadas en el balance de la estabilidad del plazo y de la mano de obra, para asegurar de no sobrepasar la máxima fecha de compleción tolerable y minimizar los costos.

V.3.3. *En Relación a la Calidad:*

- **Generación de Errores** (*ErroresPotDetec, ErroresAnalisis, TotalErroresGen*)

En el Capítulo anterior cuando se definió en el modelo propuesto de la etapa de análisis el nivel *"Requisitos Inadecuados"* se indicó que acumula una cantidad de especificaciones que se utilizarán como una medida de la calidad de las especificaciones que afecta la cantidad de código defectuoso que se produce en las fases subsiguientes de diseño y codificación. Ésta variable juntamente con la "*Funcionalidad Inicial*" (función cociente) definen la tasa de generación de errores de análisis.

Por otra parte, el modelo base al definir su límite, deja claro que excluye el desarrollo de los requisitos porque asume que no habrán subsiguientes cambios o modificaciones en los requerimientos del sistema.

Bajo estas dos consideraciones se realiza la siguiente ejecución tomando las variables: ErroresPotDetec, GeneraErrAnalis. La primera representa los errores generados en el proyecto sin contemplar los errores de análisis, es decir sólo errores de diseño y codificación. La segunda variable, indica la cantidad generada de errores de análisis

propiamente dichos y que se corrigen en esta fase a través del ciclo "*Gestión de Requisitos*".

Una primera ejecución, cuya gráfica se presenta a continuación en la Figura V.5, muestra los errores potencialmente detectables generados en el proyecto pero sin contemplar la fracción de éstos que se corrigen en la etapa de análisis. En la Figura V.5 se puede observar que esta variable alcanza un valor de 586,85 errores.

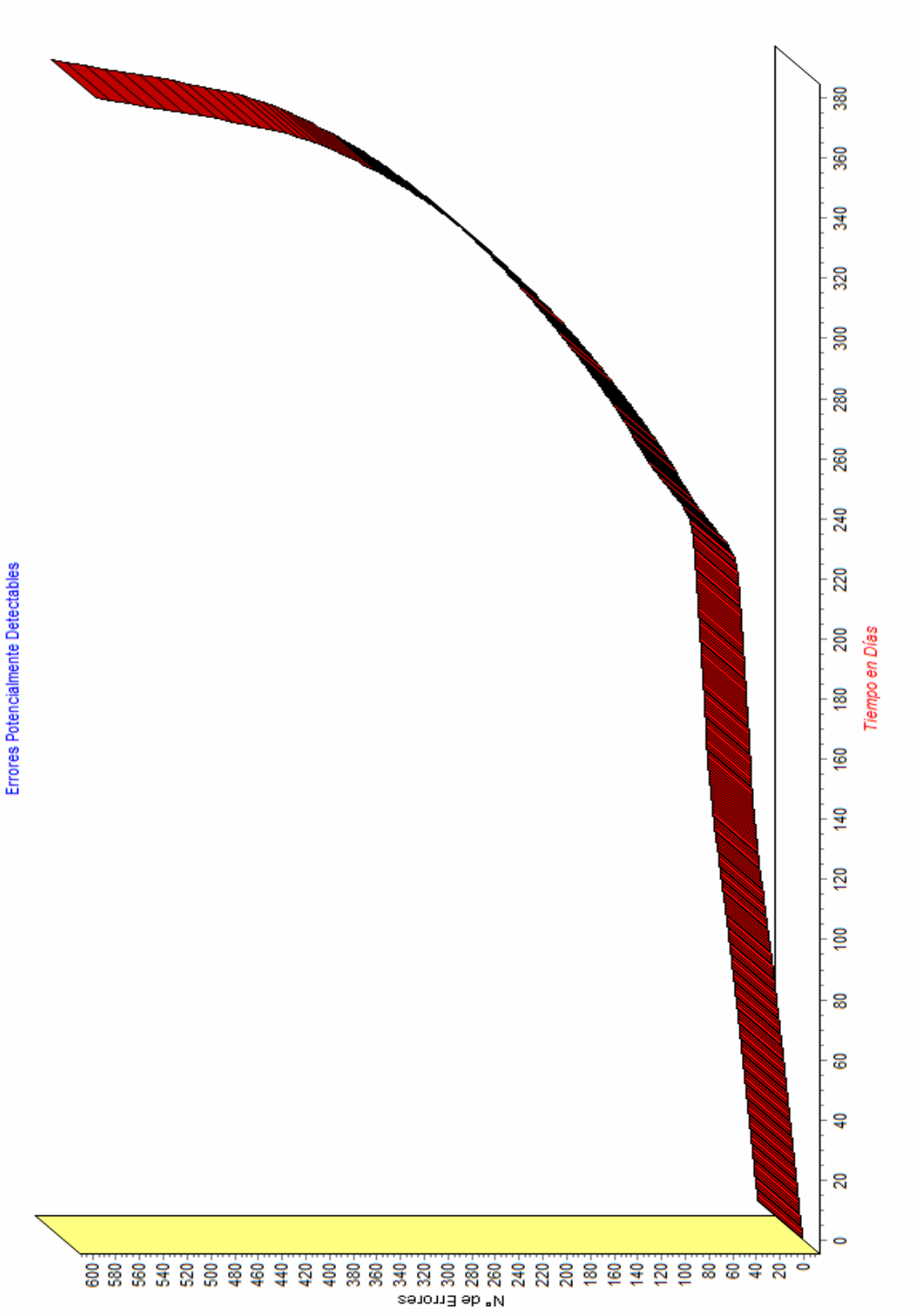


Figura V.5. Evolución temporal de la variable "Errores Potencialmente Detectables"- Con errores de análisis

Sin embargo, la simulación para la misma variable pero considerando que la proporción de errores de análisis fue corregida en esta fase, muestra una disminución importe en la cantidad total de errores potencialmente detectables. El resultado según esta ejecución, que se muestra en la Figura V.6, se reduce a 191,41 errores.

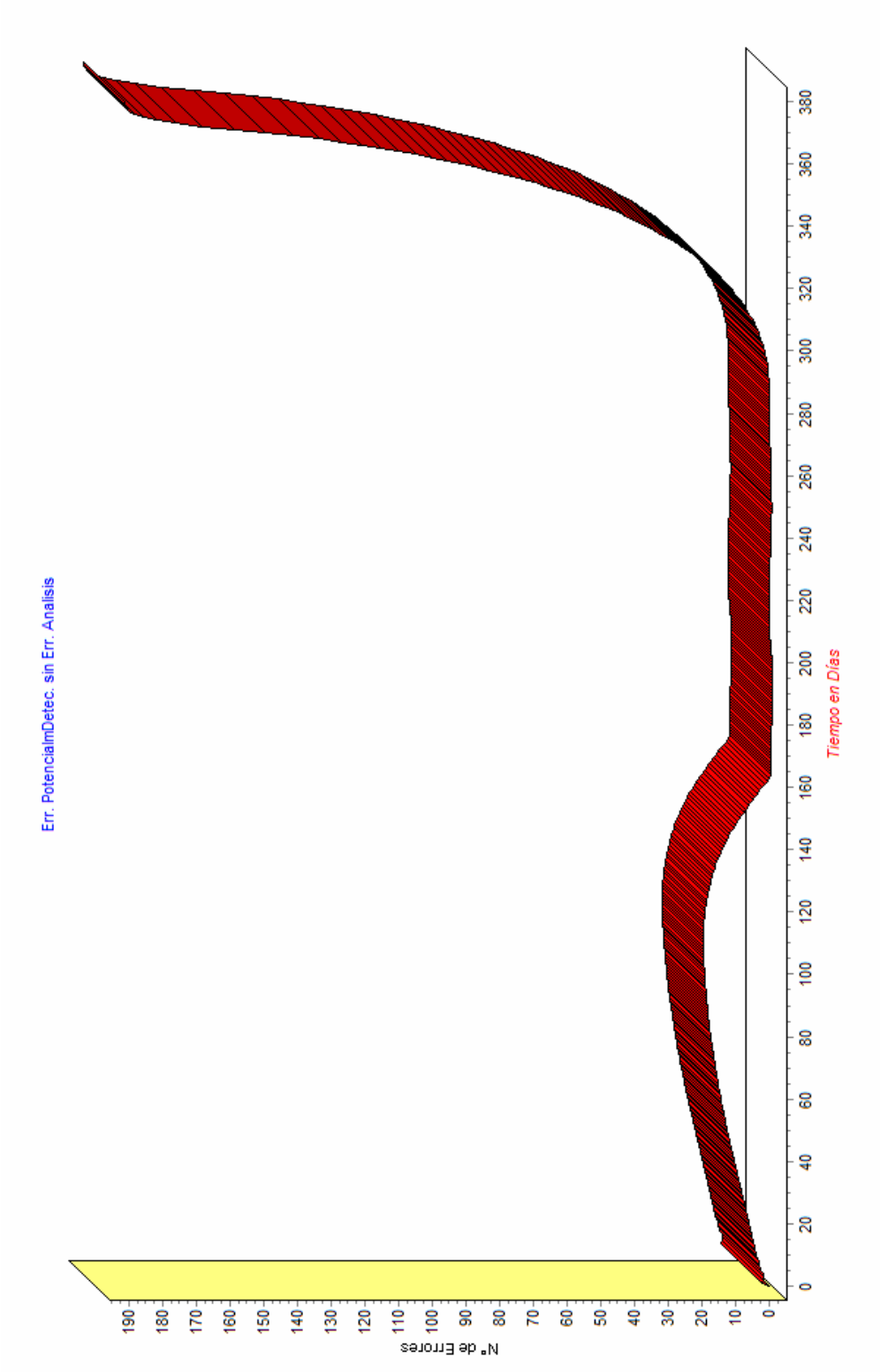


Figura V.6. Evolución temporal de la variable "Errores Potencialmente Detectables"- Sin errores de análisis

V.3.3. *En Relación a la Productividad:*

- **Porcentaje de Tareas Desarrolladas** (*PorTrabDesarPer*)

Las siguientes dos ejecuciones del modelo, que se muestras en las Figuras V.7 y V.8, se corresponden con la evolución temporal de la variable "*PorTrabDesarPer*" que mide el porcentaje de tareas desarrolladas para la simulación del modelo base y del modelo integrado, respectivamente.

Comparando estos comportamientos se puede visualizar que al cabo del mismo tiempo (380 días, que es el tiempo de simulación y tiempo real de duración del proyecto), se completa el 87% de las tareas del proyecto con el modelo base, mientras que con el modelo extendido, se completa el 94% de las tareas.

Por los resultados expuestos para esta variable y teniendo en cuenta la reducción de errores por la acción del ciclo de gestión de requisitos, probada en el item V.3.3, es válido señalar que el modelo extendido propuesto permite aumentar la productividad de desarrollo como consecuencia de la reducción del trabajo de prueba y rework que se gana por iniciar las etapas de diseño y subsiguientes, con menor carga de errores que fueron descartados en la etapa de análisis.

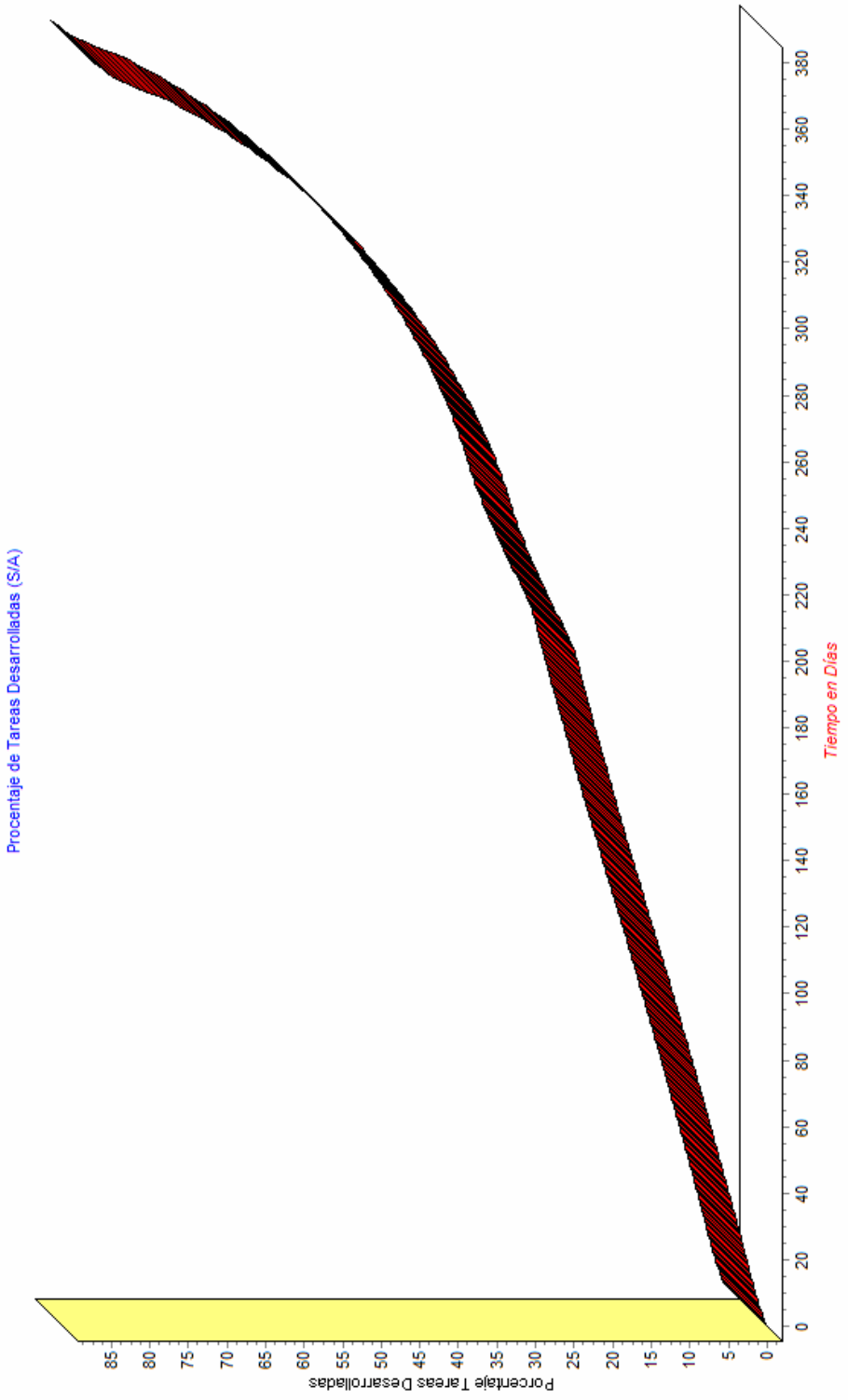


Figura V.7. Trayectoria de la variable “Porcentaje de Tareas Desarrolladas”- Con Modelo Base

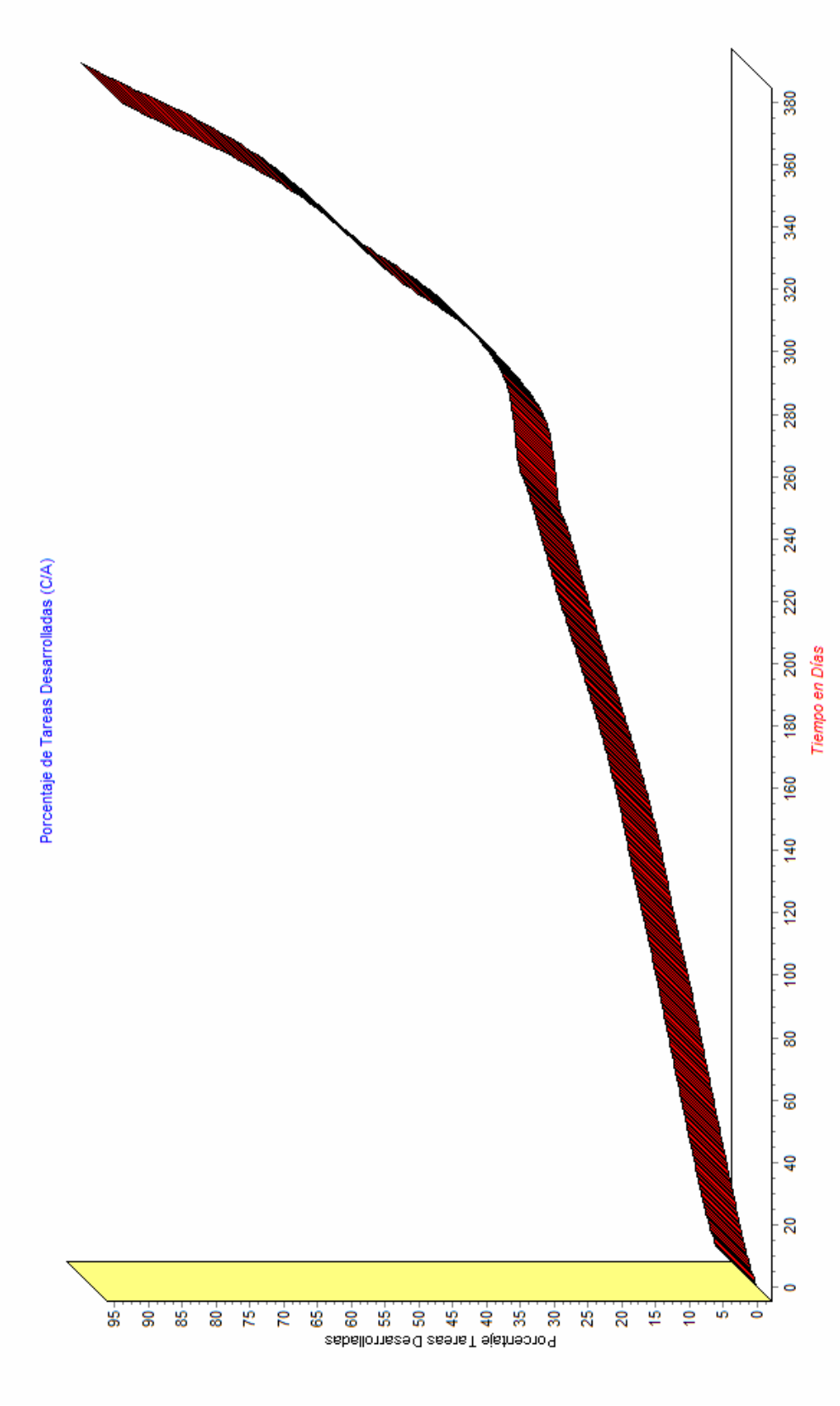


Figura V.8. Trayectoria de la variable "Porcentaje de Tareas Desarrolladas"- Con Modelo Extendido

Si bien el modelo extendido fue bastante preciso en reproducir el comportamiento dinámico del proyecto bajo estudio, éste subestimó ligeramente el valor absoluto del *esfuerzo* del proyecto en un 4%. Esta subestimación ocasionó una sobrestimación pero sólo por menos de 1% (0,61) en la *duración* del proyecto. En relación a la *calidad*, medida a través de la cantidad de errores, es evidente la reducción de un modelo a otro, y esto está en estrecha relación con la acción del ciclo de gestión de requisitos incluido en el modelo de la etapa de análisis, añadido al modelo base. Con respecto a la *productividad*, también se puedo demostrar que los resultados arrojados por el modelo extendido contribuyen a mejorar el proceso de desarrollo.

Concluyendo, luego de la ejecución del modelo extendido los valores de salida para los parámetros de entrada relacionados con el *tiempo* y *esfuerzo*, se contrastaron con los valores reales registrados para el proyecto caso de estudio. Esta comparación muestra que los resultados arrojados por el modelo propuesto son más próximos a los valores reales, que los resultados arrojados por el modelo base. Luego, se verifican ***P1:*** *El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de estimación y* ***P2:*** *El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de planificación.*

Por otra parte, aunque no se tienen registros reales para las variables *calidad* y *productividad,* analizando sus valores de salida para ambos modelos, se puedo apreciar la bondad del modelo propuesto sobre el modelo base. Entonces, se verifica ***P3:*** *El MoDEGePS ayuda a cuantificar con mayor precisión en la tarea de seguimiento y control.*

A partir de ello se puede afirmar que el modelo extendido propuesto responde positivamente a la comprobación de la hipótesis planteada en el trabajo ***H****: El MoDEGePS ayuda a cuantificar con mayor precisión en las tareas de gestión de los proyectos de desarrollo de software.*

También, para los interrogantes enunciados al principio de este Capítulo, pueden corresponder respuestas positivas.

Las ejecuciones del modelo extendido para analizar el comportamiento de estas variables arrojaron los resultados que se resumen en la siguiente Tabla V.3 y se disponen conjuntamente con los resultados arrojados por el modelo base.

Variable	Registro Real	Modelo Base	Modelo Extendido
Duración (Días)	380	387,50	382,32
Esfuerzo (Hombres/Días)	2222	2092	2129,73
Calidad (Errores)	-----	586,85	191,41
Productividad (% Tareas)	-----	87,30	94

Tabla V.3. Registro del comportamiento de las variables

Con esta disposición se puede apreciar que el modelo propuesto tiene la capacidad de predecir el comportamiento de un proyecto software, con mayor precisión que el modelo base.

CONCLUSIONES

En este trabajo se ha desarrollado un modelo de dinámica de sistemas alternativo, complementario e integrador - y la herramienta de simulación derivada del primero - que aborda el proceso de desarrollo de software desde la etapa de análisis hasta la etapa de prueba, para dar soporte a la toma de decisiones en las tareas de gestión, considerando los aspectos técnicos y socio-cultural-políticos de la organización. La herramienta se orienta a mejorar la capacidad del proceso de desarrollo para generar productos de mayor calidad.

En concreto se desarrolló en una *primera* etapa, el *modelo de gestión extendido* que a través de una interfase acopla un modelo ya existente a un modelo propuesto de la etapa de análisis; y en una *segunda* etapa se aplica un *caso de estudio* para probar el modelo global con la ejecución bajo la *herramienta de simulación*, derivada de este modelo, para estudiar y predecir las implicaciones dinámicas de un conjunto de decisiones y procedimientos de gestión.

El modelo extendido

El modelo extendido integra el conocimiento de los micro componentes de la gestión de proyectos software (programación, productividad, planificación, control), que se incluyen en el modelo escogido como base, el de Abdel-Hamid y Madnick, con el conocimiento ganado, a través de este trabajo, sobre los beneficios de incorporar los modelos del proceso de Ingeniería de Requisitos a los modelos de soporte a la gestión.

Esta integración se alcanza a través una interfase que acopla el modelo existente (que aborda el proceso de desarrollo de software desde la etapa de diseño hasta la etapa de prueba) con el modelo propuesto de la etapa de análisis. Este acople se realiza a través de las variables en común entre los modelos: *esfuerzo total, productividad de desarrollo, tareas desarrolladas, errores de análisis, requisitos inadecuados.*

El modelo extendido proporciona un enfoque integrado que ayuda a alcanzar un conocimiento global sobre el proceso de gestión. Este enfoque integrado es útil tanto para diagnosticar el problema como para evaluar soluciones en el marco de la gestión de proyectos de desarrollo de software, ya que el modelo identifica mecanismos de feedback y los usa para estructurar y clarificar relaciones en la gestión de proyectos software. Entonces, como la intervención de la gestión con frecuencia lleva a consecuencias de segundo y tercer orden, este modelo extendido con su enfoque integrado resulta útil como soporte a la toma de decisiones, porque facilita la búsqueda de múltiples conjuntos de

factores que al interactuar causan problemas en el proceso y producto software y sugiere posibles soluciones, evaluando el sistema como una totalidad socio-técnico.

El caso de estudio

Luego que el modelo se desarrolló bajo las convenciones de la Dinámica de Sistemas y se implementó como una herramienta de simulación bajo el entorno de la aplicación Evolution, se condujo la simulación del caso de estudio DE-A con el objetivo de probar la capacidad del modelo extendido para reproducir la dinámica del proceso de gestión de un proyecto software terminado, con mayor precisión que el modelo base.

El proyecto DE-A fue seleccionado como caso de estudio por satisfacer los siguientes criterios: (1) Proyecto de mediana envergadura, es decir tamaño medio (entre 16-64 KSDI) y (2) Proyecto "típico", es decir desarrollado en un entorno familiar. También por tener disponible el registro de los valores de tamaño, duración y mano obra.

Para simular el proyecto DE-A, el modelo fue primero parametrizado. Este proceso involucró establecer valores para 21 parámetros que representan el entorno particular del proyecto. Es decir, los 21 valores de parámetros no involucran cambios en la formulación de las estructuras políticas del modelo, sino que definen el entorno dentro del cual las políticas se ejercen. Esto es significativo dado que el comportamiento dinámico es mayoritariamente el resultado de la interacción de las estructuras políticas del modelo.

Los resultados de la simulación muestran que el modelo extendido fue bastante preciso en la reproducción de la historia de desarrollo real del proyecto DE-A, y más específicamente, reproduce el comportamiento dinámico de la *duración*, el *esfuerzo*, la *calidad* y la *productividad* del proyecto con mayor precisión que el modelo base.

La Experimentación

El modelo se usó como vehículo de experimentación para analizar la precisión con la cual puede predecir el comportamiento de un proyecto software. Tres aspectos fueron estudiados: *Duración, Esfuerzo, Calidad* y *Productividad.*

En relación a la variable *Duración*, expresada en la unidad de medida *días*, el experimento con el modelo extendido arrojó el valor **382,32**, un resultado mayor, aunque sólo por pocas unidades, al valor real del tiempo de duración del proyecto, **380 días**. Comparado con el resultado que arroja la ejecución con el modelo base (**387,5**) se puede

apreciar que el modelo extendido propuesto proporciona una medida de la duración más próxima a la real.

Con respecto a la variable *Esfuerzo* (medida en hombres/días), el resultado de la simulación del modelo propuesto indica un consumo total (al finalizar el proyecto) de 2129,73 hombres/días de esfuerzo. Por su parte la misma ejecución con el modelo base, proporciona como salida 2092 hombres/días. Ambos son levemente menores que el valor real registrado para el proyecto (2222 hombres/días) pero el resultado de la ejecución del modelo propuesto es más cercano al valor real.

En referencia a la variable *Calidad*, medida en función del número de errores, se demostró la bondad del ciclo de gestión de requisitos incluido en modelo extendido propuesto, ya que permitió la considerable reducción del número de errores de 586,85 a 191,41.

Finalmente, en relación a la variable *Productividad*, comparando los comportamientos de los modelos base y extendido se pudo visualizar que al cabo del mismo tiempo (380 días, que es el tiempo de simulación y tiempo real de duración del proyecto), se completa el 87% de las tareas del proyecto con el modelo base, mientras que con el modelo extendido, se completa el 94% de las tareas.

De esta manera, los resultados de todas las ejecuciones muestran que el *Modelo* integrado propuesto es capaz de proporcionar medidas que permiten cuantificar con mayor precisión en las tareas de gestión de un proyecto de desarrollo de software.

REFERENCIAS

[1] Abdel-Hamid, T. K. - Madnick, S. E. - "Software Project Dynamics. An Integrated Approach" - Prentice-Hall, New Jersey, 1991.

[2] Álvarez, S. - "Un modelo de calidad para una empresa de software" - Evento Calidad 2003 - Convención Informática - C. Habana, 2003.

[3] Aracil, Javier - Gordillo Francisco - "Dinámica de sistemas" - Alianza Editorial - Madrid, 1997.

[4] Asociación Astic - "Sólo Requisitos, un evento para la calidad del software" Disponible en: http://www.astic.es/Asociacion/DetallesBoletic/Documents/Boletic2046/astic_4.pdf

[5] Boddu, R. - "From Requirements to Testing in a Natural Way" - 12th IEEE International Requirements Engineering Conference (RE'04) - Kyoto. Japón, 2004.

[6] Brooks, F. - "No Silver Bullet: Essence and Accidents of Software Engineering" - IEEE Computer - Vol. 20, Nº 4, 1987.

[7] Burgues, Dave - Rogers, Ted - Mushrush, Chris - Kester, Dave - "Using Software Metrics and Measurements for Earned Value Toolkit" – Disponible en: https://acc.dau.mil/CommunityBrowser.aspx - Publicación: 10/2004 -

[8] Capers, Jones - "Estimatics Software Costs" - Mc Graw Hill - 2004.

[9] Dumke, R. - Schmietendorf, A. - Zuse, H. - "Formal Descriptions of Software Measurement and Evaluation. A Short Overview and Evaluation" - Otto-von-Guericke - Universität Magdeburg, Magdeburg, 2005.

[10] Ernesto, A - Lagarda, L. - "Introducción a la Dinámica de Sistemas" - Disponible en www.itson.mx/dii/elagrada/apagina2001/dinamica/powerpoint/diapositiva.ppt

[11] Evolution 3.5 - Manual de Usuario - Grupo Simon - Universidad Industrial de Santander - Colombia, 2003.

[12] Ferreira, M. J. - Loucopoulos, P. - "Organisation of analysis patterns for effective re-use" - Proceedings of the International Conference on Enterprise Information Systems - Setubal, Portugal, 2001.

[13] Glass, R. L. - "Software Engineering: Facts and Fallacies" - Addison-Wesley, 2002.

[14] González Martínez, Diana Laura - "Procedimiento para capturar requerimientos en el desarrollo de sistemas de soporte a la decisión utilizando modelación de la dinámica de sistemas - Monterrey, 2004.

[15] González Saravia, Fernando - "Modelos aplicables a la gerencia de proyectos" - Informe II Congreso Internacional de Gerencia de Proyectos – Disponible en http://www.boletines.latinpyme.com.co/latinpyme33.htm

[16] Kolde, Chris - "Basic metrics for requirements management" - Disponible en: http://www.borlan.com - Publicación: 2004

[17] Lakey, Peter B. - "A Hybrid Software Process Simulation Model for Project Management", Cognitive Concepts, 2003.

[18] Leffingwell, Dean - Widrig, Don. - "Managing Software Requirements". Second Edition. Addison-Wesley, 2003.

[19] McKeen, J. D. - "An Empirical Investigation of the Process and Product of Application System Development" - Ph. D. Dissertation - University of Minnesota, 1991.

[20] Martínez, Silvio - Requena, Alberto - "Simulación dinámica por ordenador" - Alianza Editorial - Madrid, 1988.

[21] Material Bibliográfico de la Maestría "Ingeniería del Software e Ingeniería del Conocimiento" - Facultad de Informática - Universidad Politécnica de Madrid - España.

[22] Mcdonald Landazuri, Bárbara A. - "Definición de Perfiles en Herramientas de Gestión de Requisitos" - Disponible en http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Mcdonald.pdf - Publicación: 09/2005

[23] Mohanty S. N. - "Software Cost Estimation: Present and Future". Software Practice and Experience - Vol. 11, 1981.

[24] Navarro, Antonio - "Ingeniería de Software: Ingeniería de Requisitos" - Disponible en http://www.fdi.ucm.es/profesor/anavarro/6._Ingeniería_de_requisitos.pdf

[25] Phalp, Keith - Cox, K. - "Using Enactable Models to Enhance Use Case Descriptions", 2003.

[26] Pressman, R. - "Ingeniería de Software. Un enfoque práctico" - Cuarta Edición - Mc Graw Hill, 1998.

[27] Ramos, I. - Ruiz, M. - "Modelo Dinámico simplificado para la gestión de proyectos software" - Disponible en http://www.lsi.us.es/docs/informes/mdr.pdf - Publicación: 2001

[28] Ramos, I. - Ruiz, M. - Toro, M. - "Los Modelos Dinámicos y la Ingeniería del Software" - Disponible en http://www.sc.ehu.es/jidacoj/remis/docs/modelos.htm - Publicación: 11/1999

[29] Ramos, Isabel - Aroba, Javier - Pérez, Pedro - "Los modelos dinámicos aplicados en la gestión de proyectos de desarrollo de software" - Disponible en http://www.sc.ehu.es/jidacoj/remis/docs/irr-2000.htm

[30] Rodrigues, Alexandre G. - "System Dynamics in Proyect Management: Assessing the impacts of client behavior on project performance" - Glasgow, 1997.

[31] Ruiz, M - Ramos, I. - "Estimación del Coste de la Calidad del Software a través de la Simulación del Proceso de Desarrollo" - Disponible en http://www.unab.edu.co/editorialunab/revistas/rcc/r21_art6_c.pdf

[32] Ruiz, M. - Ramos, I - "Sistema de métricas para la obtención de información de seguimiento de proyectos software" - Disponible en http://www.aeipo/congreso/2000-1/pdf/DFO2.pdf

[33] SWEBOK - "Guide to the Software Engineering Body of Knowledge" - IEEE Computer Society, 2004.

[34] The Standish Group International - "The CHAOS Report" (1994) - Disponible en http://www.standishgroup.com

[35] Tovar Caro, Edmundo - Material Bibliográfico de la Maestría "Ingeniería del Software e Ingeniería del Conocimiento" - Unidad 8: Educación de requisitos y análisis del problema - Facultad de Informática - Universidad Politécnica de Madrid - España.

ANEXOS

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Mano de obra recien contratada	WFNEW = WFNEW + (HIRERT - ASIMRT - NEWTRR) WFNEW = 0	Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software
Mano de obra experimentada	WFEXP = WFEX + (ASIMRT - EXPTRR - QUITRT) WFEXP = WFSTRT	Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software
Tasa de contratacion	HIRERT = MAX (0, WFGAP / HIREDY)	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Retraso de contratos	HIREDY = 40	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Tasa de asimilacion	ASIMRT = WFNEW / ASIMDY	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Retraso promedio de asimilacion	ASIMDY = 80	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Tasa de retiro	QUITRT = WFEXP / AVEMPT	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Tiempo promedio de empleo	AVEMPT = 673	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Mano de obra diaria para entrenamiento	DMPTRN = WFNEW * TRPNHR	Subsistema Gestion de Recursos Humanos	Sector Asignacion de Mano de Obra
Entrenadores por contratado reciente	TRPNHR = 0,2	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Mano de obra experimentada full time equivalente	FTEXWF = WFEXP * ADMPPS	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Mano de obra diaria promedio por empleado		Subsistema Gestion de Recursos Humanos	Sector Asignacion de Mano de Obra, Subsistema Planificacion
Tope de nuevos contratos	CELNWH = FTEXWF * MNHPXS	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Maximo de empleados nuevos manejable por empleado experimentado	MNHPXS = 3	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Tasa de transferencia de mano de obra recien contratada	NEWTRR = MIN (TRNFRT, WFNEW / DT)	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Retraso de transferencia	TRNFRT = MAX (0, -WFGAP / TRNSDY) TRNSDY = 10	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Tasa de transferencia de mano de obra experimentada	EXPTRR = MIN (WFEXP / DT, TRNFRT - NEWTRR)	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬
Total de mano de obra	TOTWF = WFNEW + WFEXP	Subsistema Gestion de Recursos Humanos	Sector Asig.M.de Obra, Subsect.Productiv.de Desar.de Soft.,Subsist.Planific.
Hueco de mano de obra	WFGAP = WFS - TOTWF	Subsistema Gestion de Recursos Humanos	¬¬¬¬¬

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Tope de total de mano de obra	CELTWF = CELNWH + WFEXP	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Nivel de mano de obra buscada	WFS = MIN (CELTWF, WFNEED)	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Nivel de mano de obra necesaria		Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Porcentaje de trabajo realizado		Sector Asignacion de Mano de Obra	Subsec.Prod.de Desar.de Soft,Sector G.de Calid.y Rew.,Sector Prueba del Sist.
Mano de obra diaria promedio por empleado	ADMPPS = 1	Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos, Subsistema Planificacion
Total de mano de obra		Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos, Subsistema Planificacion
Total de mano de obra diaria	TOTDMP = TOTWF * ADMPPS	Sector Asignacion de Mano de Obra	Subsector Productiv. de Desar. de Soft., Subsist. Planificacion
Fraccion planeada de mano de obra para QA	PFMPQA = TABHL (TPFMQA, PJBAWK, 0, 1, .1) * (1 + QO / 100) QO = O	Sector Asignacion de Mano de Obra	¬¬¬¬¬
Presion de tiempo		Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework, Subsistema Control
Fraccion real de mano de obra para QA	AFMPQA = PFMPQA * (1 + ADJQA) AFMPQA = PFMPQA	Sector Asignacion de Mano de Obra	¬¬¬¬¬
Mano de obra diaria para entrenamiento		Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos
Mano de obra diaria disponible despues del entrenamiento	DMPATR = TOTDMP - DMPTRN	Sector Asignacion de Mano de Obra	¬¬¬¬¬
Mano de obra de rework real necesaria		Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Mano de obra diaria para QA	DMPQA = MIN ((AFMPQA * TOTDMP), .9 * DMPATR)	Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Mano de obra diaria para produccion de software	DMPSWP = DMPATR - DMPQA	Sector Asignacion de Mano de Obra	¬¬¬¬¬
Mano de obra de rework percibida necesaria por error	PRWMPE = PRWMPE + (DT / TARMPE)(RWMPPE - PRWMPE) PRWMPE = .5 TARMPE = 10	Sector Asignacion de Mano de Obra	Subsistema Control
Mano de obra diaria para rework	DMPRW = MIN ((DESECR * PRWMPE), DMPSWP) DMPRW = 0	Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Mano de obra diaria para desarrollo y prueba	DMPDVT = DMPSWP - DMPRW	Sector Asignacion de Mano de Obra	Sector Desarrollo de Software
Tasa de correccion de error deseada	DESECR = DTCERR / DESRWD DESECR = 0	Sector Asignacion de Mano de Obra	¬¬¬¬¬

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Retraso de rework deseado	DESRWD = 15	Sector Asignacion de Mano de Obra	¬¬¬¬¬
Errores descubiertos / Errores detectados	DTCERR o DSCERR en el diagrama	Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Tamaño de trabajo percibido		Sector Desarrollo de Software	Subsistema Control
Mano de obra diaria para desarrollo y prueba		Sector Desarrollo de Software	Sector Asignacion de Mano de Obra
Fraccion de esfuerzo para prueba del sistema	FREFTS = TABHL (TFEFTS, TSKPRM / PJBSZ, 0, .2, .04)	Sector Desarrollo de Software	¬¬¬¬¬
Tareas percibidas restantes		Sector Desarrollo de Software	Subsistema Control, Ajuste del Tamaño del Trabajo
Mano de obra diaria para desarrollo de software	DMPSDV = DMPDVT * (1 - FREFTS)	Sector Desarrollo de Software	¬¬¬¬¬
Tasa de desarrollo de software	SDVRT = MIN ((DMPSDV * SDVPRD), TSKPRM / DT) SDVRT = 0	Sector Desarrollo de Software	Sector Garantia de Calidad y Rework, Sector Prueba del Sistema
Tareas desarrolladas / Tareas trabajadas		Sector Desarrollo de Software	Sector Garantia de Calidad y Rework
Productividad de desarrollo de software	SDVPRD = POTPRD * MPDMCL	Sector Desarrollo de Software	Subsector Productividad de Desarrollo de Software
Mano de obra recien contratada		Subsector Productividad de Desarrollo de Software	Subsistema Gestion de Recursos Humanos
Mano de obra experimentada		Subsector Productividad de Desarrollo de Software	Subsistema Gestion de Recursos Humanos
Total de mano de obra		Subsector Productividad de Desarrollo de Software	Sector Asig.de M.Obra,Subsist.Gest.RRHH de Desar.de Soft.,Subsist.Planific.
Productividad potencial normal de empleados experimentados	NPWPEX = 1	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Productividad potencial normal de empleados recien contratados	NPWPNE = 0.5	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Comunicacion entre empleados (overhead)	COMMOH = TABHL (TCOMOH, TOTWF, 0, 30, 5)	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Fraccion de mano de obra experimentada	FRWFEX = WFEXP / TOTWF Definida en Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Productividad potencial normal promedio	ANPPRD = FRWFEX * NPWPEX + (1 - FRWFEX) * NPWPNE	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Productividad de desarrollo de software	SDVPRD = POTPRD * MPDMCL	Subsector Productividad de Desarrollo de Software	Sector Desarrollo de Software

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Multiplicador de productividad debido a perdidas	MPDMCL = AFMDPJ * (1 - COMMOH)	Subsector Productividad de Desarrollo de Software	Sector Garantia de Calidad y Rework, Sector Prueba del Sistema
Productividad potencial	POTPRD = ANPPRD * MPPTPD	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Porcentaje de trabajo realizado		Subsector Productividad de Desarrollo de Software	Sect.Asig. MO.,de Soft,Sector Garantia de Cal.y Rew.,Sector Prueba del Sist.
Tiempo de retraso para reducir el cansancio	EXHDDY = 20	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Tasa de vaciamiento/reduccion del nivel de cansancio	RDEXHL = CLIP (EXHLEV / EXHDDY, 0, 0, RIEXHL)	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Cansancio	EXHLEV = EXHLEV + DT * (RIEXHL - RDEXHL) EXHLEV = 0	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Fraccion normal de esfuerzo sobre el proyecto	NFMDPJ = .6	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Fraccion real de esfuerzo sobre el proyecto	AFMDPJ = AFMDPJ + DT * WRADJR AFMDPJ = NFMDPJ	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Tasa de incremento del nivel de cansancio	RIEXHL = TABHL (TRIXHL, (1 - AFMDPJ) / (1 - NFMDPJ), -0.5, 1, .1)	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬
Tiempo de ajuste a la tasa de trabajo	WKRADY = NWRADY * EWKRTS NWRADY = TABHL (TNWRAD, TIMERM, 0, 30, 5) EWKRTS = CLIP (1, .75, WKRTS, AFMDPJ)	Subsector Productividad de Desarrollo de Software	¬¬¬¬¬

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Cansancio	EXHLEV = EXHLEV + DT * (RIEXHL - RDEXHL) EXHLEV = 0	Subsector Productividad de Desarrollo de Software	
Comunicacion entre empleados (overhead)	COMMOH = TABHL (TCOMOH, TOTWF, 0, 30, 5)	Subsector Productividad de Desarrollo de Software	
Densidad de error	ERRDSY = ANERPT * 1000 / DSIPTK	Sector Garantia de Calidad y Rework	
Densidad de error		Sector Prueba del Sistema	Sector Garantia de Calidad (se calcula con otras variables)
Densidad de error activo	AERRDS = UDAVER / (CUMTQA + .1)	Sector Prueba del Sistema	
Densidad de error alisada	MAERED = TABHL (TMERED, SMOOTH (AERRDS * 1000/ DSIPTK, TSAEDS), 0, 100, 10) TSAEDS = 40 Multiplicador de la regeneracion de error activo debido a la densidad de error, figura en las abreviatura	Sector Prueba del Sistema	
Densidad de error pasivo	PERRDS = UDPVER / (CUMTQA + .0001)	Sector Prueba del Sistema	
Disponibilidad para cambiar mano de obra	WCWF = MAX (WCWF1, WCWF2)	Subsistema Planificacion	
Disponibilidad para cambiar mano de obra 1	WCWF1 = TABHL (TWCWF1, TIMERM / (HIREDY + ASIMDY), 0, 3, .3) TWCWF1 = 0 / 0 / .1 / .4 / . 85 / 1 / 1 / 1 / 1 / 1 / 1	Subsistema Planificacion	
Disponibilidad para cambiar mano de obra 2	WCWF2 = TABHL (TWCWF2, SCHCDT / MXTLCD, .86, 1, .02) TWCWF2 = 0 / .1 / .2 / .35 / .6 / .7 / .77 / .80	Subsistema Planificacion	
Entrenadores por contratado reciente	TRPNHR = 0,2	Subsistema Gestion de Recursos Humanos	
Errores activos no detectados	UDAVER = UDAVER + DT * (AEGRT + AERGRT - AERRRT - DCRTAE) UDAVER = 0	Sector Prueba del Sistema	
Errores corregidos	CMRWED = CMRWED + DT * RWRATE CMRWED = 0	Sector Garantia de Calidad y Rework	
Errores descubiertos	DTCERR o DSCERR en el diagrama	Sector Asignacion de Mano de Obra	
Errores detectados	DTCERR = DTCERR + DT * (ERRDRT - RWRATE) DTCERR = 0	Sector Garantia de Calidad y Rework	
Errores detectados		Subsistema Control	Sector Garantia de Calidad y Rework
Errores normales cometidos por tarea	NERPTK = NERPK * DSIPTK / 1000	Sector Garantia de Calidad y Rework	
Errores pasivos no detectados	UDPVER = UDPVER + DT * (PEGRT + AERRRT - DCRTPE) UDPVER = 0	Sector Prueba del Sistema	
Errores potencialmente detectables	PTDTER = PTDTER + DT * (ERRGRT - ERRDRT - ERRSRT) PTDTER = 0	Sector Garantia de Calidad y Rework	
Escasez percibida en esfuerzo		Subsector Productividad de Desarrollo de Software	
Escasez/Exceso de esfuerzo informado	SHRRPT = PMDSHR - MDHDL	Subsistema Control	
Escasez/Exceso de esfuerzo percibido	PMDSHR = TMDPSN - MDRM	Subsistema Control	

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Esfuerzo de desarrollo acumulado		Subsistema Control	
Esfuerzo de prueba acumulado		Subsistema Control	
Esfuerzo gastado acumulado		Ajuste del Tamaño del Trabajo	
Esfuerzo informado todavia necesario	MDRPTN = MDRM + SHRRPT	Subsistema Control	Ajuste del Tamaño del Trabajo
Esfuerzo informado todavia necesario		Ajuste del Tamaño del Trabajo	Subsistema Control
Esfuerzo manejado	MDHDL = CLIP (MIN (MAXSHR, PMDSHR), EXSABS, PMDSHR, 0) * CTRLSW CTRLSW = 1 Esfuerzo q sera manejado (escasez) o absorbido (exceso)	Subsector Productividad de Desarrollo de Software	Subsistema Control
Esfuerzo manejado		Subsistema Control	Subsector Productividad de Desarrollo de Software
Esfuerzo para desarrollo		Subsistema Control	Dice que ya esta en otro y no esta
Esfuerzo para prueba		Subsistema Control	
Esfuerzo percibido necesario para rework errores detectados	MDPNRW = DTCERR * PRWMPE	Subsistema Control	
Esfuerzo percibido restante para nuevas tareas	PJDPRD = TSKPRM / (MDPRNT + .1)	Subsistema Control	Ajuste del Tamaño del Trabajo
Esfuerzo percibido restante para nuevas tareas		Ajuste del Tamaño del Trabajo	Subsistema Control
Esfuerzo percibido todavia necesario para nuevas tareas	MDPNNT = TSKPRM / ASSPRD	Subsistema Control	
Esfuerzo percibido todavia necesario para prueba	MDPNTS = TSTPRM / PRTPRD	Subsistema Control	
Esfuerzo restante		Subsector Productividad de Desarrollo de Software	Subsistema Control, Ajuste del Tamaño del Trabajo, Subsistema Planificacion
Esfuerzo restante		Subsistema Control	Ajuste del Tamaño del Trabajo
Esfuerzo restante	MDRM = MAX (.0001, JBSZMD - CUMMD)	Ajuste del Tamaño del Trabajo	
Esfuerzo restante		Subsistema Planificacion	Subsect. Produc. Desar. Soft., Sector Gar. Cal. y Rew., Sector Prueba Sist., Ajte. Tamaño Trab.

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Esfuerzo total percibido todavia necesario		Subsector Productividad de Desarrollo de Software	Subsistema Control Subsector Productividad de Desarrollo de Software
Esfuerzo total percibido todavia necesario	TMDPSN = MDPNNT + MDPNTS + MDPNRW	Subsistema Control	
Exceso percibido en esfuerzo	EXSABS = MAX (0, (TABHL (TEXABS, TMDPSN / MDRM, 0, 1, .1) * MDRM - TMDPSN))	Subsector Productividad de Desarrollo de Software	
Fecha de complecion indicada	INDCDT = TIME + TIMEPR	Subsistema Planificacion	
Fecha de complecion programada	SCHCDT = SCHCDT + DT * (INDCDT - SCHCDT) / SCHADT SCHCDT = TDEV	Subsistema Planificacion	
Fraccion de esfuerzo para prueba del sistema	FREFTS = TABHL (TFEFTS, TSKPRM / PJBSZ, 0, .2, .04)	Sector Desarrollo de Software	
Fraccion de mano de obra experimentada	FRWFEX = WFEXP / TOTWF Definida en Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software	
Fraccion de retiro	AERRFR = TABHL (TERMFR, PJBAWK, 0, 1, .1)	Sector Prueba del Sistema	
Fraccion de tareas adicionales agregadas al esfuerzo	FADHWO = TABHL (TFAHWO, RSZDCT / (MSZTWO + .001), 0, 2, .2) TFAHWO = 0 / 0 / 0 / 0 / 0 / 0 / .7 / .9 / .975 / 1 / 1	Ajuste del Tamaño del Trabajo	
Fraccion normal de esfuerzo sobre el proyecto	NFMDPJ = .6	Subsector Productividad de Desarrollo de Software	
Fraccion planeada de mano de obra para QA	PFMPQA = TABHL (TPFMQA, PJBAWK, 0, 1, .1) * (1 + QO / 100) QO = O	Sector Asignacion de Mano de Obra	
Fraccion real de esfuerzo sobre el proyecto	AFMDPJ = AFMDPJ + DT * WRADJR AFMDPJ = NFMDPJ	Subsector Productividad de Desarrollo de Software	
Fraccion real de mano de obra para QA	AFMPQA = PFMPQA * (1 + ADJQA) AFMPQA = PFMPQA	Sector Asignacion de Mano de Obra	
Hueco de mano de obra	WFGAP = WFS - TOTWF	Subsistema Gestion de Recursos Humanos	
Mano de obra de prueba necesaria por tarea	TMPNPT = (TSTOVH * DSIPTK / 1000 + TMPNPE * (PERRDS + AERRDS)) / MPDMCL	Sector Prueba del Sistema	
Mano de obra de prueba normal por error	TMPNPE = .15 No dice normal en la abreviatura, dice necesaria	Sector Prueba del Sistema	
Mano de obra de QA necesaria para detectar un error	QAMPNE = NQAMPE * (1 / MPDMCL) * MDEFED	Sector Garantia de Calidad y Rework	
Mano de obra de QA normal necesaria por error	NQAMPE = TABHL (TNQAPE, PJBAWK, 0, 1, .1)	Sector Garantia de Calidad y Rework	
Mano de obra de rework normal necesaria por error	NRWMPE = TABHL (TNRWME, PJBAWK, 0, 1, .2)	Sector Garantia de Calidad y Rework	
Mano de obra de rework percibida necesaria por error	PRWMPE = PRWMPE + (DT / TARMPE)(RWMPPE - PRWMPE) PRWMPE = .5 TARMPE = 10	Sector Asignacion de Mano de Obra	Subsistema Control Sector Asignación de Mano de Obra
Mano de obra de rework percibida necesaria por error		Subsistema Control	
Mano de obra de rework real necesaria		Sector Asignacion de Mano de Obra	

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Mano de obra de rework real necesaria por error	RWMPPE = NRWMPE / MPDMCL No dice real en la abreviatura	Sector Garantia de Calidad y Rework	
Mano de obra diaria disponible despues del entrenamiento	DMPATR = TOTDMP - DMPTRN	Sector Asignacion de Mano de Obra	
Mano de obra diaria para desarrollo de software	DMPSDV = DMPDVT * (1 - FREFTS)	Sector Desarrollo de Software	
Mano de obra diaria para desarrollo y prueba	DMPDVT = DMPSWP - DMPRW	Sector Asignacion de Mano de Obra	Sector Desarrollo de Software
Mano de obra diaria para desarrollo y prueba		Sector Desarrollo de Software	Sector Asignacion de Mano de Obra
Mano de obra diaria para entrenamiento	DMPTRN = WFNEW * TRPNHR	Subsistema Gestion de Recursos Humanos	Sector Asignacion de Mano de Obra
Mano de obra diaria para entrenamiento		Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos
Mano de obra diaria para produccion de software	DMPSWP = DMPATR - DMPQA	Sector Asignacion de Mano de Obra	
Mano de obra diaria para prueba	DMPTST = DMPDVT * FREFTS Definida en este Sector	Sector Prueba del Sistema	
Mano de obra diaria para QA	DMPQA = MIN ((AFMPQA * TOTDMP), .9 * DMPATR)	Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Mano de obra diaria para QA		Sector Garantia de Calidad y Rework	Sector Asignacion de Mano de Obra
Mano de obra diaria para rework	DMPRW = MIN ((DESECR * PRWMPE), DMPSWP) DMPRW = 0	Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework
Mano de obra diaria para rework		Sector Garantia de Calidad y Rework	Sector Asignacion de Mano de Obra
Mano de obra diaria promedio por empleado		Subsistema Gestion de Recursos Humanos	Sector Asignacion de Mano de Obra, Subsistema Planificacion
Mano de obra diaria promedio por empleado	ADMPPS = 1	Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos, Subsistema Planificacion
Mano de obra diaria promedio por empleado		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos, Sector Asignación de Mano de Obra

VARIABLE	*ABREVIATURA EN DYNAMO*		*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Mano de obra experimentada	WFEXP = WFEX + (ASIMRT - EXPTRR - QUITRT)	WFEXP = WFSTRT	Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software
Mano de obra experimentada			Subsector Productividad de Desarrollo de Software	Subsistema Gestion de Recursos Humanos
Mano de obra experimentada full time equivalente	FTEXWF = WFEXP * ADMPPS		Subsistema Gestion de Recursos Humanos	
Mano de obra full time equivalente			Subsector Productividad de Desarrollo de Software	
Mano de obra recien contratada	WFNEW = WFNEW + (HIRERT - ASIMRT - NEWTRR)	WFNEW = 0	Subsistema Gestion de Recursos Humanos	Subsector Productividad de Desarrollo de Software
Mano de obra recien contratada			Subsector Productividad de Desarrollo de Software	Subsistema Gestion de Recursos Humanos
Maxima escasez de esfuerzo a ser manejado	MAXSHR = (OVWDTH * FTEQWF * MAXMHR) * WTOVWK		Subsector Productividad de Desarrollo de Software	
Maxima fecha de complecion tolerable	MXTLCD = MXSCDC * TDEV		Subsistema Planificacion	
Maximo ajuste en horas/hombre	MAXMHR = 1		Subsector Productividad de Desarrollo de Software	
Maximo cansancio tolerable	MXEXHT = 50		Subsector Productividad de Desarrollo de Software	
Maximo de empleados nuevos manejable por empleado experimentado	MNHPXS = 3		Subsistema Gestion de Recursos Humanos	
Maximo tamaño relativo de las adiciones toleradas sin agregar esfuerzo al proyecto	MSZTWO = .01		Ajuste del Tamaño del Trabajo	
Multiplicador de productividad debido a perdidas	MPDMCL = AFMDPJ * (1 - COMMOH)		Subsector Productividad de Desarrollo de Software	Sector Garantia de Calidad y Rework, Sector Prueba del Sistema
Multiplicador de productividad debido a perdidas			Sector Prueba del Sistema	Subsector Productividad de Desarrollo de Software
Multiplicador de productividad debido a perdidas (2 veces en el diagrama)			Sector Garantia de Calidad y Rework	Subsector Productividad de Desarrollo de Software
Multiplicador de productividad potencial debido al aprendizaje	MPPTPD = TABHL (TMPTPD, PJBAWK, 0, 1, .1)		Subsector Productividad de Desarrollo de Software	
Multiplicador de umbral de duracion de sobrecarga debido al cansancio	MODTEX = TABHL (TMODEX, EXHLEV / MXEXHT, 0, 1, .1)		Subsector Productividad de Desarrollo de Software	
Nivel de mano de obra buscada	WFS = MIN (CELTWF, WFNEED)		Subsistema Gestion de Recursos Humanos	Subsistema Planificacion

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Nivel de mano de obra buscada		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Nivel de mano de obra indicada	WFINDC = MDRM / (TIMERM + .001)) / ADMPPS	Subsistema Planificacion	
Nivel de mano de obra necesaria		Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Nivel de mano de obra necesaria	WFNEED = MIN ((WCWF * WFINDC + (1 - WCWF) * TOTWF), WFINDC)	Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Peso a la productividad proyectada	WTPJDP = MPWDEV * MPWREX	Subsistema Control	
Porcentaje de ajuste en la tasa de trabajo buscado	WKRTS = (1 + PBWKRS) * NFMDPJ PBWKRS = CLIP ((MDHDL / (FTEQWF * (OVWDTH + .0001))), (MDHDL / (TMDPSN - MDHDL + .0001)), PMDSHR, 0)	Subsector Productividad de Desarrollo de Software	
Porcentaje de arreglos malos	PBADFX = .075 Definida en Sector Prueba del Sistema	Sector Garantia de Calidad y Rework	
Porcentaje de errores activos	FRAERR = TABHL (TFRAER, PJBAWK, 0, 1, .1) Fraccion de errores escapados q seran activos, se define en abreviat.	Sector Prueba del Sistema	
Porcentaje de mano de obra experimentada = Fraccion de mano de obra ex	FRWFEX = WFEXP / TOTWF Esta definic vale si es "Fraccion…MO Exp." Definida en Subsist. Gestion de RR HH	Sector Garantia de Calidad y Rework	Subsector Productividad de Desarrollo de Software (si es fraccion de mano de obra ex.)
Porcentaje de tareas no descubiertas q se descubren por dia	PUTDPD = TABHL (TPUTDD, PJBPWK, 0, 100, 20) TPUTDD = 0 / 0.4 / 2.5 / 5 / 10 / 100	Ajuste del Tamaño del Trabajo	
Porcentaje de trabajo desarrollado percibido	PDEVRC = SMOOTH (MAX ((100 - ((MDRPTN - MDPNTS) / (JBSZMD - TSSZMD)) * 100), PDEVRC), RPTDLY) PDEVRC = 0 En las abreviaturas figura como "% de desarrollo percibido completo"	Subsistema Control	Ajuste del Tamaño del Trabajo
Porcentaje de trabajo desarrollado percibido	PJBPWK = (CMTKDV / PJBSZ) * 100 Porcentaje de trabajo percibido realizado figura en las abreviaturas	Ajuste del Tamaño del Trabajo	
Porcentaje de trabajo realizado		Sector Asignacion de Mano de Obra	Subsector Productiv. de Desar. de Soft, Sector Garantia de Cal. y Rew., Sector Prueba del Sist.
Porcentaje de trabajo realizado		Subsector Productividad de Desarrollo de Software	
Porcentaje de trabajo realizado		Sector Prueba del Sistema	
Porcentaje de trabajo realizado (2 veces en el diagrama)		Sector Garantia de Calidad y Rework	

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Presion de tiempo		Sector Asignacion de Mano de Obra	Sector Garantia de Calidad y Rework, Subsistema Control
Presion de tiempo		Sector Garantia de Calidad y Rework	Subsistema Control
Productividad de desarrollo asumida	ASSPRD = PJDPRD * WTPJDP + PRDPRD * (1 - WTPJDP)	Subsistema Control	Ajuste del Tamaño del Trabajo
Productividad de desarrollo asumida (2 veces)		Ajuste del Tamaño del Trabajo	Subsistema Control
Productividad de desarrollo de software	SDVPRD = POTPRD * MPDMCL	Sector Desarrollo de Software	Subsector Productividad de Desarrollo de Software
Productividad de desarrollo de software	SDVPRD = POTPRD * MPDMCL	Subsector Productividad de Desarrollo de Software	Sector Desarrollo de Software
Productividad de desarrollo percibida	PRDPRD = CMTKDV / (CUMMD - CMTSMD)	Subsistema Control	
Productividad de desarrollo proyectada	PJDPRD = TSKPRM / (MDPRNT + .1)	Subsistema Control	
Productividad de prueba percibida	PRTPRD = SMOOTH ((CLIP (PLTSPD, ACTSPD, 0 , CUMTKT)), TSTSPD)	Subsistema Control	
Productividad de prueba percibida		Ajuste del Tamaño del Trabajo	Subsistema Control
Productividad de prueba planeada	PLTSPD = PJBSZ / TSSZMD	Subsistema Control	
Productividad de prueba real	ACTSPD = CUMTKT / (CMTSMD + .001)	Subsistema Control	
Productividad potencial	POTPRD = ANPPRD * MPPTPD	Subsector Productividad de Desarrollo de Software	
Productividad potencial normal de empleados experimentados	NPWPEX = 1	Subsector Productividad de Desarrollo de Software	
Productividad potencial normal de empleados recien contratados	NPWPNE = 0.5	Subsector Productividad de Desarrollo de Software	
Productividad potencial normal promedio	ANPPRD = FRWFEX * NPWPEX + (1 - FRWFEX) * NPWPNE	Subsector Productividad de Desarrollo de Software	
Prueba entre empleados (overhead)	TSTOVH = 1 Esfuerzo de prueba overhead	Sector Prueba del Sistema	
Retraso de contratos	HIREDY = 40	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Retraso de contratos		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Retraso de rework deseado	DESRWD = 15	Sector Asignacion de Mano de Obra	
Retraso de transferencia	TRNFRT = MAX (0, -WFGAP / TRNSDY) TRNSDY = 10	Subsistema Gestion de Recursos Humanos	
Retraso en el ajuste del tamaño del trabajo en esfuerzo	DAJBMD = TABHL (TDAJMD, TIMERM, 0, 20, 20) TDAJMD = .5 / 3	Ajuste del Tamaño del Trabajo	
Retraso promedio de asimilacion	ASIMDY = 80	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Retraso promedio de asimilacion		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Retraso promedio de incorporacion de tareas descubiertas	DLINCT = 10	Ajuste del Tamaño del Trabajo	
Retraso promedio de QA	AQADLY = 10	Sector Garantia de Calidad y Rework	
Tamaño de trabajo percibido		Sector Desarrollo de Software	
			Sera igual a Tamaño de trabajo percibido" del Sector Desarrollo de software?
Tamaño de trabajo percibido en tareas		Subsistema Control	
Tamaño del trabajo en tareas percibido actualmente	PJBSZ = PJBSZ + DT * RTINCT PJBSZ = PJBDSI / DSIPTK	Ajuste del Tamaño del Trabajo	
Tamaño percibido de tareas descubiertas en esfuerzo	PSZDCT = TKDSCV / ASSPRD	Ajuste del Tamaño del Trabajo	
Tamaño relativo de las tareas descubiertas	RSZDCT = PSZDCT / (MDPRNT + .0001)	Ajuste del Tamaño del Trabajo	
Tamaño total del trabajo en esfuerzo	JBSZMD = JBSZMD + DT * (IRDVDT + IRTSDT + ARTJBM) JBSZMD = DEVMD + TSTMD	Ajuste del Tamaño del Trabajo	
Tareas de trabajo no descubiertas	UNDJTK = UNDJTK - DT * RTDSTK UNDJTK = RJBSZ - PJBSZ	Ajuste del Tamaño del Trabajo	
Tareas desarrolladas / Tareas trabajadas		Sector Desarrollo de Software	Sector Garantia de Calidad y Rework
Tareas desarrolladas / Tareas trabajadas	TSKWK = TSKWK + DT * (SDVRT - QART) TSKWK = 0	Sector Garantia de Calidad y Rework	Sector Desarrollo de Software
Tareas desarrolladas acumuladas	CMTKDV = CMTKDV + DT * SDVRT CMTKDV = 0	Subsistema Control	
Tareas desarrolladas acumuladas		Ajuste del Tamaño del Trabajo	Dice que ya esta y no esta
Tareas descubiertas = Tareas QA (Revisadas) Acumuladas	CUMTQA = CUMTQA + DT * (QART - TSRATE) CUMTQA = 0 Esta definicion vale si es Tareas QA'ed Acumu.	Sector Garantia de Calidad y Rework	Sector Garantia de Calidad y Rework si es Tareas QA (revisadas) acumuladas
Tareas descubiertas = Tareas QA (Revisadas) Acumuladas	TKDSCV = MAX ((TKDSCV + DT * (RTDSTK - RTINCT)), 0) TKDSCV = 0	Ajuste del Tamaño del Trabajo	
Tareas percibidas restantes		Sector Desarrollo de Software	Subsistema Control, Ajuste del Tamaño del Trabajo
Tareas percibidas restantes		Subsistema Control	Ajuste del Tamaño del Trabajo

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Tareas percibidas restantes	TSKPRM = PJBSZ - CMTKDV "Nuevas tareas percibidas restantes" figura en las abreviaturas	Ajuste del Tamaño del Trabajo	
Tareas probadas acumuladas	CUMTKT = CUMTKT + DT * TSRATE CUMTKT = 0	Sector Prueba del Sistema	Subsistema Control
Tareas probadas acumuladas (2 veces)		Subsistema Control	Sector Prueba del Sistema
Tareas QA (revisadas)		Sector Prueba del Sistema	Sector Garantia de Calidad y Rework
Tareas restantes por probar	TSTPRM = PJBSZ - CUMTKT	Subsistema Control	
Tasa de ajuste de la tasa de trabajo	WRADJR = (WKRTS - AFMDPJ) / WKRADY	Subsector Productividad de Desarrollo de Software	
Tasa de ajuste de plazo	No esta definida en las abreviaturas	Subsistema Planificacion	
Tasa de ajuste del tamaño del trabajo en esfuerzo	ARTJBM = (MDRPTN + CUMMD - JBSZMD) / DAJBMD	Ajuste del Tamaño del Trabajo	
Tasa de asimilacion	ASIMRT = WFNEW / ASIMDY	Subsistema Gestion de Recursos Humanos	
Tasa de contratacion	HIRERT = MAX (0, WFGAP / HIREDY)	Subsistema Gestion de Recursos Humanos	
Tasa de correccion de error deseada	DESECR = DTCERR / DESRWD DESECR = 0	Sector Asignacion de Mano de Obra	
Tasa de desarrollo de software	SDVRT = MIN ((DMPSDV * SDVPRD), TSKPRM / DT) SDVRT = 0	Sector Desarrollo de Software	Sector Garantia de Calidad y Rework, Sector Prueba del Sistema
Tasa de desarrollo de software		Sector Garantia de Calidad y Rework	Sector Desarrollo de Software
Tasa de desarrollo de software		Sector Prueba del Sistema	Sector Desarrollo de Software
Tasa de descubrimiento de tareas	RTDSTK = UNDJTK * PUTDPD / 100	Ajuste del Tamaño del Trabajo	
Tasa de deteccion de error	ERRDRT = MIN (PERDRT, PTDTER / DT)	Sector Garantia de Calidad y Rework	
Tasa de deteccion de error potencial	PERDRT = DMPQA / QAMPNE	Sector Garantia de Calidad y Rework	
Tasa de deteccion y correccion de errores activos	DCRTAE = MIN (TSRATE * AERRDS, UDAVER / DT) Si, corresponde agregar …de errores activos q esta en rojo	Sector Prueba del Sistema	
Tasa de deteccion y correccion de errores pasivos	DCRTPE = MIN (TSRATE * PERRDS, UDPVER / DT) Si, corresponde agregar …de errores pasivos q esta en rojo	Sector Prueba del Sistema	
Tasa de escape de error	ERRSRT = QART * ANERPT	Sector Garantia de Calidad y Rework	
Tasa de escape de error		Sector Prueba del Sistema	Sector Garantia de Calidad y Rework
Tasa de escape de error y de generacion de arreglos malos	Es la suma de ERRSRT + BDFXGR y se usa en la definicion de tasa de generacion activo / pasivo	Sector Prueba del Sistema	

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Tasa de generacion de arreglos malos	BDEXGR = RWRATE * PBADFX Definida en Sector Prueba del Sistema	Sector Garantia de Calidad y Rework	
Tasa de generacion de arreglos malos		Sector Prueba del Sistema	Sector Garantia de Calidad y Rework
Tasa de generacion de error	ERRGRT = SDVRT * ERRPTK	Sector Garantia de Calidad y Rework	
Tasa de generacion de error activo	AEGRT = (ERRSRT + BDFXGR) * FRAERR	Sector Prueba del Sistema	
Tasa de generacion de error pasivo	PEGRT = (ERRSRT + BDFXGR) * (1 - FRAERR)	Sector Prueba del Sistema	
Tasa de incorporacion de tareas descubiertas al proyecto	RTINCT = DELAY3 (RTDSTK, DLINCT)	Ajuste del Tamaño del Trabajo	
Tasa de incremento del nivel de cansancio	RIEXHL = TABHL (TRIXHL, (1 - AFMDPJ) / (1 - NFMDPJ), -0.5, 1, .1)	Subsector Productividad de Desarrollo de Software	
Tasa de incremento en el esfuerzo de desarrollo debido a tareas descubiertas	IRDVDT = (RTINCT / ASSPRD) * FADHWO	Ajuste del Tamaño del Trabajo	
Tasa de incremento en la prueba debido a tareas descubiertas	IRTSDT = (RTINCT / PRTPRD) * FADHWO)	Ajuste del Tamaño del Trabajo	
Tasa de prueba	TSRATE = MIN (CUMTQA / DT, DMPTST / TMPNPT)	Sector Prueba del Sistema	
Tasa de QA	QART = DELAY3 (SDVRT, AQADLY)	Sector Garantia de Calidad y Rework	
Tasa de regeneracion de error activo	AERGRT = SDVRT * SMOOTH (AERRDS, TSAEDS) * MAERED	Sector Prueba del Sistema	
Tasa de retiro	QUITRT = WFEXP / AVEMPT	Subsistema Gestion de Recursos Humanos	
Tasa de retiro de errores activos	AERRRT = UDAVER * AERRFR	Sector Prueba del Sistema	
Tasa de rework	RWRATE = DMPRW / RWMPPE	Sector Garantia de Calidad y Rework	
Tasa de transferencia de mano de obra experimentada	EXPTRR = MIN (WFEXP / DT, TRNFRT - NEWTRR)	Subsistema Gestion de Recursos Humanos	
Tasa de transferencia de mano de obra recien contratada	NEWTRR = MIN (TRNFRT, WFNEW / DT)	Subsistema Gestion de Recursos Humanos	
Tasa de vaciamiento/reduccion del nivel de cansancio	RDEXHL = CLIP (EXHLEV / EXHDDY, 0, 0, RIEXHL)	Subsector Productividad de Desarrollo de Software	
Tiempo	No esta definida en las abreviaturas	Subsistema Planificacion	
Tiempo de ajuste a la tasa de trabajo	WKRADY = NWRADY * EWKRTS NWRADY = TABHL (TNWRAD, TIMERM, 0, 30, 5) EWKRTS = CLIP (1, .75, WKRTS, AFMDPJ)	Subsector Productividad de Desarrollo de Software	
Tiempo de ajuste de plazo	SCHADT = TABHL (TSHADT, TIMERM, 0, 5, 5) TSHADT = .5 / 5	Subsistema Planificacion	
Tiempo de retraso para reducir el cansancio	EXHDDY = 20	Subsector Productividad de Desarrollo de Software	
Tiempo percibido todavia restante	TIMEPR = MDRM / (WFS * ADMPS) "Tiempo percibido todavia requerido" figura en las abreviaturas	Subsistema Planificacion	
Tiempo promedio de empleo	AVEMPT = 673	Subsistema Gestion de Recursos Humanos	

VARIABLE	*ABREVIATURA EN DYNAMO*	*DIAGRAMA PRINCIPAL*	*DIAGRAMA SECUNDARIO*
Tiempo restante	TIMERM = MAX (SCHCDT - TIME, 0)	Subsistema Planificacion	
Tope de nuevos contratos	CELNWH = FTEXWF * MNHPXS	Subsistema Gestion de Recursos Humanos	
Tope de total de mano de obra	CELTWF = CELNWH + WFEXP	Subsistema Gestion de Recursos Humanos	Subsistema Planificacion
Tope de total de mano de obra		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Total de mano de obra	TOTWF = WFNEW + WFEXP	Subsistema Gestion de Recursos Humanos	Sector Asig. de Mano de Obra, Subsector Productiv. de Desar. de Soft., Subsist. Planificacion
Total de mano de obra		Sector Asignacion de Mano de Obra	Subsistema Gestion de Recursos Humanos, Subsistema Planificacion
Total de mano de obra		Subsector Productividad de Desarrollo de Software	Subsistema Gestion de Recursos Humanos
Total de mano de obra		Subsistema Planificacion	Subsistema Gestion de Recursos Humanos
Total de mano de obra diaria	TOTDMP = TOTWF * ADMPPS	Sector Asignacion de Mano de Obra	
Umbral de duracion de la sobrecarga	OVWDTH = NOVWDT * MODTEX	Subsector Productividad de Desarrollo de Software	
Umbral de duracion de sobrecarga normal	NOVWDT = TABHL (TNOWDT, TIMERM, 0, 50, 10)	Subsector Productividad de Desarrollo de Software	