Universidad Nacional de La Plata
Facultad de Informática

Diseño de un Simulador Dinámico de Proyectos de Desarrollo de Software que utilizan Metodología Scrum

Tesis presentada para obtener el grado de Magister en Ingeniería de Software

Autor: Diego Alberto Godoy

Director: Dr. Eduardo O. Sosa
Co-Director: Dr. Gustavo Rossi

2014

Contenido

Lista de Figuras

Lista de Tablas

Glosario

Arquetipo	"El mejor de su clase"
Bucle de retroalimentación	Son las estructuras básicas de los sistemas que determinan realmente la dinámica del comportamiento de los sistemas. E un bucle de retroalimentación (feedback) (o lazo realimentado) cuando en un diagrama de influencias existe una cadena de influencias circulares cerradas
BurdownChart	El BurdownChart es un tablero que permite realizar diversas simulaciones: ver cómo se aplazan las fechas de entrega si se le añaden requisitos, ver cómo se avanzan si se le quitan requisitos o se añade otro equipo, etc.
Ciclo de Vida	Describe el desarrollo de software, desde la fase inicial hasta la fase final. En Scrum el ciclo de vida se divide en 3 fases denominadas: Pregame, Development Game, Postgame
CMM-SW	El Modelo de Madurez de la Capacidad para el desarrollo de Software (Capability Maturity Model for Software, SW-CMM) es un modelo de procesos para el desarrollo y mantenimiento de sistemas de software
CPM	El método CPM o Ruta Crítica (equivalente a la sigla en inglés Critical Path Method) es frecuentemente utilizado en el desarrollo y control de proyectos. El objetivo principal es determinar la duración de un proyecto.
Crecimiento Sigmoidal	Presentan una estructura donde hay un proceso reforzador de crecimiento que opera por sí mismo durante un tiempo.

	Luego se encuentra con otro proceso compensador que tiende a estabilizarlo, y que opera para limitar el crecimiento
Diagrama de Forrester	Diagrama Usado para representar el comportamiento de los sistemas con un símil hidráulico.
Diagrama de Influencias	Diagrama usado para representar la estructura de los sistemas
Estructuras Básicas	Se refiere a los bucles de retroalimentación Positiva y negativa.
Feedback	Retroalimentación. Mecanismo por el cual una cierta proporción de la salida de un sistema se redirige a la entrada, con objeto de controlar su comportamiento. La realimentación permite el control de un sistema y que el mismo tome medidas de corrección con base en la información realimentada.
GERT	Graphical Evaluation and Review Technique
IPMA	International Project Management Association
Ley de Brooks	Enuncia que "Agregar mano de obra a un proyecto de software atrasado lo atrasa aún más"
Lookup	Función discreta del tipo F(x,y), donde para cada valor de "x" se obtiene el correspondiente valor asociado "y".
Método de Montecarlo	Es un método numérico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias

PERT	Program evaluation and review technique. El diagrama PERT es una representación gráfica de las relaciones entre las tareas del proyecto que permite calcular los tiempos del proyecto de forma sencilla.
Proyecto	Es un esfuerzo que se lleva a cabo para crear un producto, servicio o resultado único, y tiene la característica de ser naturalmente temporal, es decir, que tiene un inicio y un final establecidos, y que el final se alcanza cuando se logran los objetivos del proyecto o cuando se termina el proyecto porque sus objetivos no se cumplirán o no pueden ser cumplidos, o cuando ya no existe la necesidad que dio origen al proyecto
Retardo	Tiempo que pasa en se realiza una acción y su efecto.
Riesgo	La combinación de la probabilidad de que se produzca un evento y sus consecuencias negativa
SPICE	Acrónimo inglés de Simulation Program with Integrated Circuits Emphasis (Programa de simulación con énfasis en circuitos integrados)
Sprint	Bloque de tiempo iterativo en el cual se desarrolla o mejora una funcionalidad de un sistema para producir nuevos incrementos
Time-to-marked	Tiempo entre que se desarrolla un producto y que está disponible para los clientes
Vensim	Herramienta visual de modelaje que permite conceptualizar, documentar, simular, analizar y optimizar modelos de dinámica de sistemas

Resumen

En este trabajo de tesis se propone el diseño de un modelo de simulación del proceso de desarrollo de software que utiliza la Metodología de desarrollo Ágil Scrum.

Se ha realizado un análisis de la metodología Scrum, la Metodología de Dinámica de Sistemas y de trabajos similares, correspondientes a simuladores de desarrollo de proyectos de software tanto de metodologías ágiles como tradicionales.

El modelo se ha construido utilizando la metodología de dinámica de sistemas propuesta por Jay Forrester[1], que consta de la fase de Conceptualización, Formulación y Evaluación. El modelo diseñado se ha divido en los siguientes Subsistemas: Planificación, Producción, Desarrollo de Tareas, Pruebas de Desarrollo, Pruebas de Integración, Presión en el Plazo, Desarrollo de Tareas Extras, Promociones de R.H., Experiencia de R.H., Cansancio de R.H., Horas Trabajadas de R.H., Inasistencias de R.H.

Para la implementación del modelo se utilizado la herramienta Vensim PLE en versión educativa.

Para la validación del modelo se han utilizado tres casos de proyectos reales de software que siguieron la metodología Scrum. Dentro de los parámetros que se pueden establecer previo al inicio de cada simulación se encuentran: la duración y la velocidad de cada Sprint, la velocidad estimada de desarrollo de las tareas, Factores de Cansancio, de Presión en el plazo, Cantidad de integrantes del Team según su experiencia en la metodología y las tareas extras que se prevén puedan surgir. A través de la modificación de valores de los parámetros durante su ejecución el usuario puede establecer o modificar la cantidad de integrantes del Team que abandonan el proyecto, clasificar al Team mediante la asociación de estos a su experiencia en Scrum en Juniors o Expertos, cambiar la cantidad de horas estimadas de duración del proyecto, generar horas extras e inasistencia de los integrantes de manera determinística o pseudoaleatoria, entre otros.

Una vez validado el modelo se propusieron cuatro casos de experimentación, los cuales se sometieron a distintas políticas para probar las posibilidades del modelo construido frente a situaciones que se dan en los proyectos de software que utilizan Scrum para su gestión. Las políticas propuestas, simulan decisiones de los Scrum Master y el team y tienden a que los proyectos puedan completarse en el tiempo previamente estimado o por el contrario extender el tiempo para terminar con todas las tareas minimizando los errores. Otro aspecto estudiado en los experimentos es el de la gestión de los recursos humanos, en cuanto a la contratación y promoción de miembros del Team, Senior y Junior. Estos modelos fueron probados en orden creciente de dificultad y utilización de la mayor cantidad de variables intervienes.

Como conclusión se pude decir que el modelo cumple con su objetivo de ser de utilidad para el Scrum Master y el Team a la hora analizar el efecto del uso conjunto de la metodología Scrum en proyectos de desarrollo de software. Lo que diferencia a este

[1] Un mayor detalle biográfico de Jay W. Forrester se puede ver en:
http://www.ieeeghn.org/wiki/index.php/Jay_W._Forrester

trabajo de otros relacionados es que se han modelado las características esenciales de la metodología Scrum aplicada a proyectos de desarrollo de software.

La dificultad de contar con datos posmortem de proyectos de software desarrollados siguiendo scrum es una dificultad que se ha presentado a la hora de validar el modelo. Es por ello que como trabajos futuros se propone el diseño de una base de datos de proyectos de desarrollo de software que utilicen metodologías ágiles.

Por otro lado se pretende que el modelo construido pueda ser utilizado para el entrenamiento de Scrum Masters y Miembros del Team, en la estimación y gestión de proyectos que utilicen Scrum como Metodología.

Introducción

La complejidad de los sistemas de actuales, los cambios repentinos en el contexto de estos sistemas y las modificaciones en los requerimientos del cliente una vez que se ha comenzado el desarrollo de un proyecto de software, generan un ambiente donde la planificación, el desarrollo, la administración y el control del mismo resultan difíciles de estimar o evaluar.

Este tipo de escenarios requiere de metodologías de desarrollo de software que permitan generar resultados rápidamente. Entre las metodologías con este tipo de características se encuentra Scrum, la cual fue aplicada por primera vez por Ken Schwaber y Jeff Sutherland, quienes la documentaron en detalle en su obra "Agile Software Development with Scrum" [1]. Esta metodología centra su atención en las actividades de gerencia basándose principalmente en una planificación adaptativa y en el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo.

Frente a escenarios y requerimientos cambiantes, contar con una herramienta que permita simular la gestión de proyectos de desarrollo de software llevados a cabo con Scrum, representa una alternativa interesante para que los administradores puedan evaluar el impacto de sus decisiones sobre la gestión en el desarrollo del proyecto, sin influir o poner en riesgo el proyecto real y sus recursos.

Si bien actualmente existen diferentes trabajos y herramientas que permiten simular la administración de proyectos de software, como por ejemplo "Dynamics of Agile Software Development" [2], "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment" [3], "Modelo Dinámico de Simulación de Proyectos de Software con XP" [4] y [5], "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics" [6]; es necesario contar con una herramienta de simulación específica para proyectos de desarrollo de software que utilizan la metodología ágil Scrum como soporte o ayuda a los Scrum Máster y el Team en la gestión de los proyectos. En la actualidad, Scrum es de especial interés dado que es una de las metodologías ágiles más utilizadas [7]

Objetivos

Para guiar el desarrollo de este trabajo se han planteado los siguientes objetivos:

General:

- Diseñar una Herramienta de Simulación de procesos de desarrollo de software llevados a cabo con metodología Scrum utilizando Dinámica de Sistemas.

Específicos:

- Analizar la bibliografía y trabajos existentes sobre Metodología Scrum y modelos de simulación dinámicos para gestionar proyectos de desarrollo de software.

- Construir un modelo de simulación dinámico para la gestión de proyectos de desarrollo de software que utilizan Scrum, utilizando la metodología dinámica de sistemas.
- Validar el modelo con proyectos reales.
- Evaluar y validar el modelo ante modificaciones en los parámetros principales del mismo generando las conclusiones correspondientes.

Artículos presentados en Congresos, Workshops y Simposios.

Como resultado de este trababo de tesis se han presentado diversos artìculos en Congresos y Workshops y simpocios, los cuales se mencionan a continuación:

- "Simulando Proyectos de Desarrollo deSoftware Administrados con Scrum," in *XVI Workshop de Investigadores en Ciencias de la Computación RedUNCI*, Ushuaia, 2014. [8]

- "Evaluación de Alternativas de Gestión en Proyectos de Software Desarrollados con Scrum utilizando Dinámica de Sistemas," in *XX Congreso Argentina de Ciencias de la Computación RedUNCI*, San Justo, Buenos Aires, 2014. [9].

- "Simulación Dinámica de Gestión de Tareas en Proyectos Desarrollados Con Scrum" in *II Congreso Nacional de ingenieria informatica/ingenieria de sistemas (CoNaIISI). Universidad Nacional de San Luis, San Luis*, 2014. [10].

- "Modelo de Simulación Dinámico de Proyectos de Desarrollo de Software con Scrum" in *XVII Workshop de Investigadores en Ciencias de la Computación RedUNCI*, Salta, 2015 (Aceptado para Publicación y Exposición)

Estructura del Trabajo

El presente trabajo de tesis está estructurado de la siguiente manera. En Capítulo 1 se presentan los conceptos fundamentales de la simulación y se da una introducción a la metodología de dinámica de sistemas, se describen las fases en que se encuentra dividida la misma, las estructuras básicas los modelos diagramas, y los arquetipos sistémicos.

El Capítulo 2 se da una introducción a las metodologías de gestión de proyectos, en donde se explican cuestiones generales de la administración de proyectos y las metodologías existentes. Asimismo se estudian con detalle los roles, bloques de tiempo, artefactos y el ciclo de vida de la metodología Scrum.

En el Capítulo 3 se presentan trabajos relacionados de modelos de simulación dinámicos para proyectos gestionados con otras metodologías tanto ágiles como tradicionales.

A partir del Capítulo 4 se presenta el Diseño del Simulador Dinámico de Proyectos de Desarrollos de Software que utilizan Metodología Scrum. En este capítulo se

construye el modelo de la estructura del sistema, que se corresponde con el desarrollo de la fase de Conceptualización, de la metodología de dinámica de sistemas. Se utilizan los diagramas de influencias para representar la estructura del modelo de simulación propuesto en este trabajo dividido en subsistemas.

En el Capítulo 5, en base al modelo de la estructura representada por el diagrama de influencias de la Fase de Conceptualización, se presentan los modelos de comportamiento de los distintos subsistemas propuestos cuyos diseños se representan a través de los diagramas de Forrester y las ecuaciones diferenciales correspondientes a la en la Fase de Formulación de la metodología de dinámica de sistemas.

El Capítulo 6 considera la Fase de Evaluación de la metodología de dinámica de sistemas. Se presentan tres casos de validación, con datos de proyectos reales, para el modelo construido en la Fase de Formulación. Seguidamente se proponen cuatro experimentos que representan situaciones en las cuales se requieren decisiones de gestión y estimación en proyectos desarrollados con Scrum. Para cada situación se proponen políticas que pueden ser de utilidad, para resolver las mencionadas situaciones. Para finalizar este capítulo se realiza un análisis de sensibilidad de parámetros y variables destacadas del modelo.

Por último se presentaran las conclusiones y los trabajos futuros que podrían realizarse tomando como base el presente trabajo.

1 Simulación Dinámica de Sistemas

En el presente capítulo se introducirán los conceptos básicos de sistemas, modelos y simulación. Seguidamente se dará una introducción a la Dinámica de Sistemas, donde se explicaran las relaciones de influencias que interrelacionan variables, los diagramas de influencias, bucles de retroalimentación, retrasos, arquetipos, diagramas de Forrester y los distintos tipos de variables intervinientes. De la misma manera se dará una explicación de la utilización del software Vensim PLE, que será utilizado para la construcción del modelo. Finalmente se introducirán la metodología de Dinámica de Sistemas con sus respectivas fases de Conceptualización, Formulación y Evaluación.

1.1 Conceptos

La Simulación es la técnica numérica que consiste en realizar experimentos de muestreo sobre el modelo de un sistema. Un modelo no es más que un conjunto de variables junto con ecuaciones matemáticas que las relacionan y restricciones sobre dichas variables. El modelado es una etapa presente en la mayor parte de los trabajos de investigación. La mayoría de las veces, la realidad es bastante compleja como para ser estudiada directamente y es preferible la formulación de un modelo que contenga las variables más importantes que intervienen en el fenómeno en estudio y las relaciones principales entre ellas. Frecuentemente, la resolución de los problemas que se pretenden abordar puede realizarse por procedimientos analíticos sobre el modelo construido, mediante el uso de herramientas matemáticas como las de resolución de ecuaciones ordinarias o de ecuaciones diferenciales, el cálculo de probabilidades, etc. En otras circunstancias dicha resolución analítica no es posible o es extremadamente complicada y/o costosa y es preferible una aproximación de la solución mediante simulación.

Es por ello que se utiliza un modelo como medio para la experimentación en sustitución del sistema real. Los experimentos pueden llegar a tener un alto grado de sofisticación que requiera la utilización de técnicas estadísticas de diseño de experimentos. En la mayor parte de los casos los experimentos de simulación son la manera de obtener respuestas a preguntas del tipo "¿qué pasaría sí…?", preguntas cuyo objetivo suele ser evaluar el impacto de una posible alternativa que sirva de soporte a un proceso de toma de decisiones sobre un sistema, proceso que puede representarse esquemáticamente mediante el diagrama de la Figura 1.

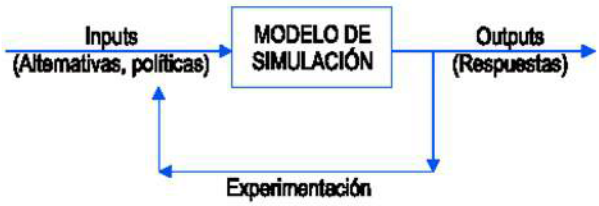


Figura 1 – Proceso de Simulación.

A continuación con fin de formalizar los conceptos presentados se darán las definiciones de los conceptos de Sistema, Modelo y Simulación.

1.1.1 Sistema

Javier Aracil en su libro Dinámica de Sistemas, define a un sistema de manera formal como "*un objeto dotado de alguna complejidad, formado por partes coordinadas, de modo que el conjunto posea una cierta unidad, que es precisamente el sistema*" [11].

En esta definición se destaca que el sistema es un conjunto de partes que interactúan y se influyen mutuamente. Este conjunto de interrelaciones generan un sistema que se presenta como único en su entorno, pudiendo preservar este, el sentido de unicidad en distintos contextos y a lo largo del tiempo.

1.1.2 Modelo

Según M. Minsky que dice: "*Un Modelo es una maqueta que permite reproducir un determinado aspecto de la realidad. Es un objeto que representa a otro. Diremos que para un observador O un M es un modelo de un objeto S -un sistema- si O puede servir de M para responder cuestiones que le importan con relación a S*" [12]. De manera resumida se puede decir que M es una herramienta que le permite al observador, representar algún aspecto que considera relevante del sistema.

1.1.3 Simulación

La simulación es la "*Técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos de tiempo*" [13].

La simulación es la técnica que imita el funcionamiento de un sistema del mundo real cuando evoluciona en el tiempo. La simulación consiste en la generación artificial del comportamiento histórico y a futuro del sistema, y la observación de ese comportamiento para hacer inferencias concernientes al funcionamiento del sistema real que estamos representando. La simulación es una metodología de resolución de para la solución de problemas del mundo real. La simulación es usada para describir y analizar el comportamiento de un sistema, responder preguntas del tipo "¿Qué pasaría si…? Y ayudar en el diseño de sistemas reales. Tanto los sistemas existentes como los conceptuales pueden ser modelados con la simulación. [14].

1.1.4 Clasificación de la simulación

Según [15], [16] y [14] la simulación se puede clasificar de la siguiente manera:

- **Estática o Dinámica:**

La simulación es estática si en el modelo el tiempo no tiene influencia en las variables de estado, mientras que es dinámica cuando el tiempo es una de las variables importantes del modelo. En la simulación estática resulta muy sencillo comparar distintas estrategias ante las mismas condiciones de azar, mientras que esto es más complicado en la simulación dinámica, exigiendo un trabajo de planificación mayor, además, el costo computacional de la simulación estática es más moderado.

La simulación estática es usada frecuentemente por los estadísticos para comprobar el comportamiento comparativo de diversos métodos estadísticos alternativos para tamaños de muestras finitos. En la simulación dinámica, normalmente se trata de ir analizando los distintos estados por los que va pasando un sistema que evoluciona en el tiempo.

- **Continua o Discreta:**

Existen dos grandes tipos de simulación dinámica: la simulación continua, en la que se supone que el sistema cambia de estado constantemente y la simulación discreta, para la cual los cambios se producen en ciertos instantes de tiempo regulares (cuantos). Sus nombres vienen dados por que en el primer caso el conjunto de estados es continuo y se pueden representar por medio de ecuaciones diferenciales como se puede ver en [11], mientras que el segundo es discreto y el seguimiento de los cambios de estado requiere la identificación de qué es lo que causa el cambio y cuando lo causa, lo que denominaremos un evento, las ecuaciones del modelo se convierten entonces en las ecuaciones y relaciones lógicas que determinan las condiciones en que tiene lugar la ocurrencia de un evento. Dentro de la simulación discreta se puede distinguir la simulación por eventos y la simulación por cuantos de tiempo.

- **Simulación por cuantos y por eventos:**

Con el nombre de simulación por eventos, o asincrónica, denominamos al tipo de simulación dinámica discreta en la cual se controla la variable tiempo moviéndola hasta la ocurrencia del siguiente evento [17]. Esto implica la necesidad de controlar minuciosamente cuál es el próximo evento, es decir saber cuáles son los posibles eventos en un futuro inmediato y cuál de ellos es el más inmediato, luego se actualiza el sistema determinando su nuevo estado, que es el resultado de este evento y generado aleatoriamente el tiempo hasta la siguiente ocurrencia de un evento de cualquier tipo que pueda ocurrir estando en este estado. También se registra la información deseada sobre el comportamiento del sistema.

La simulación por cuantos, o asincrónica, responde a una filosofía totalmente diferente. Se trata de examinar el sistema, su evolución en tiempo, dejando pasar pequeños intervalos de tiempo de longitud t, fija, llamada cuanto, en los cuales se supone que, a lo sumo, un sólo evento puede producirse.

En general, la simulación por eventos es exacta y de más difícil implementación, pero de mucha más rápida ejecución que la simulación por cuantos. Sin embargo esta

última es, muchas veces, la única posibilidad factible en la simulación dinámica continua [15] y [14].

Otra forma de clasificación de Simulación sería la siguiente:

- **Estocástica o Determinística:**

Es estocástica si alguna de las variables del modelo es una variable aleatoria, sin importar su distribución de probabilidad o por el contrario todas las variables se comportan en forma determinista.

En la siguiente Figura 2 podremos ver un resumen de los diferentes tipos de simulación.

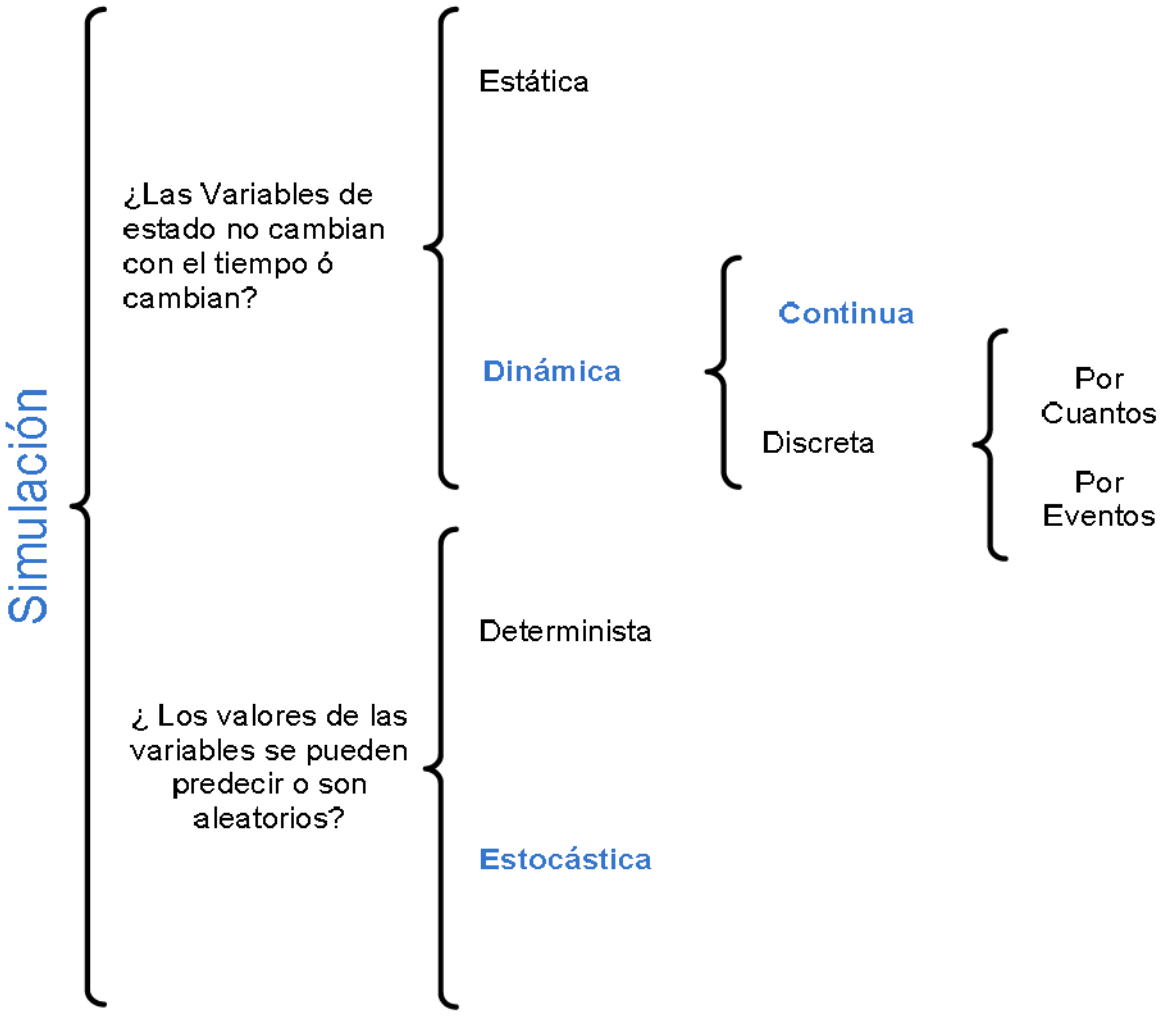


Figura 2 - Resumen de los tipos de Simulación

Se destaca en negrita el tipo de simulación utilizado en este trabajo.

1.2 Dinámica de Sistemas

La Dinámica de sistemas [11] involucra dos grandes componentes: los Sistemas y la Dinámica. La dinámica se refiere a todo lo que no es estático, es decir todo aquello que cambia constantemente, cuando se menciona a la dinámica de un sistema se refiere a que todas las variables que conforman al mismo sufren modificaciones o cambios a lo

largo del tiempo, ya sea, influenciada por los cambios de otras variables o por factores externos capaces de provocar un cambio en las mismas.

Este conjunto de elementos que se interrelacionan de manera dinámica son los que en cierta medida provocan los cambios en el sistema. Los cambios en un sistema se manifiestan mediante su comportamiento y, la trama de relaciones constituye lo que se denomina su estructura. Así, se pone de manifiesto que la dinámica de sistemas intenta reflejar como un todo la relación existente entre el comportamiento de un sistema y la estructura del mismo.

Es así que es posible representar a un sistema de manera sencilla a partir de un conjunto de elementos que lo componen *C* y por un conjunto de relaciones de influencias *R* que establece cómo se producen las influencias entre las partes.

Tanto los componentes como las relaciones pueden representarse mediante un grafo orientado cuyos nodos representan componentes y cuyas aristas o flechas representan las relaciones de influencia entre los componentes, a las cuales se adicionan símbolos positivos a las flechas (+) para representar una relación de influencia positiva de un objeto A sobre otro B, esto es, al incrementarse A, también lo hace B y símbolos negativos (-) a las flechas, cuando la relación de influencia es negativa de un objeto A sobre otro B, esto es, al modificar su valor A, B lo hace en sentido opuesto.

Un diagrama de influencias básico se observa en la Figura 3.

Figura 3 - Diagrama Influencia Básico [11]

1.2.1 Estructuras Básicas de los Sistemas

Las estructuras básicas de los sistemas son los bucles de retroalimentación, los cuales determinan realmente la dinámica del comportamiento de los sistemas.

Se dice que existe un bucle de retroalimentación (feedback) (o lazo realimentado) cuando en un diagrama existe una cadena de influencia circulares cerradas. Conviene distinguir dos tipos de lazos realimentados, lazos positivos y negativos.

Así como existen relaciones de influencia positivas y negativas también existen bucles de retroalimentación positiva y negativa [11]. Adicionalmente a las estructuras básicas existen los denominados retrasos que se explicaran en la sección 1.2.3.

1.2.2 Bucle Realimentación Negativa

Un Bucle de Realimentación negativa, es una estructura que reacciona e intenta anular una influencia o perturbación externa a algunos de los objetos componentes del mismo. Es por esta razón que al bucle de realimentación negativa se lo considera una

estructura estabilizadora del sistema, ya que, busca que el mismo mantenga su estructura ante alguna perturbación.

En la Figura 4(a) se presenta un bucle de retroalimentación negativa A, B, C. Donde A influye positivamente sobre B, y, a su vez, B influye positivamente sobre C. Pero C influye negativamente sobre A

Analizando detalladamente el conjunto de influencias, en la primera de ellas, un incremento de la variable A hará que B se incremente, ya que existe una influencia positiva de la primera sobre la segunda variable. Igual situación ocurrirá con C, al incrementarse B. Por el contrario un incremento de C, hará que A disminuya. Esto se debe a que la influencia que presenta es negativa. Finalmente, al disminuir A, se producirá un decremento de B tendiendo así a anular el incremento inicial que sufriera. Adicionalmente el bucle de retroalimentación negativa suele contar con una constante cuyo valor indica el valor en el que el sistema tendería a estabilizarse.

En la Figura 4(b) se presenta el comportamiento que tendría la variable C.

Una manera sencilla de identificar a un Bucle de Realimentación Negativa, es a partir de que el número de influencias negativa sea impar.

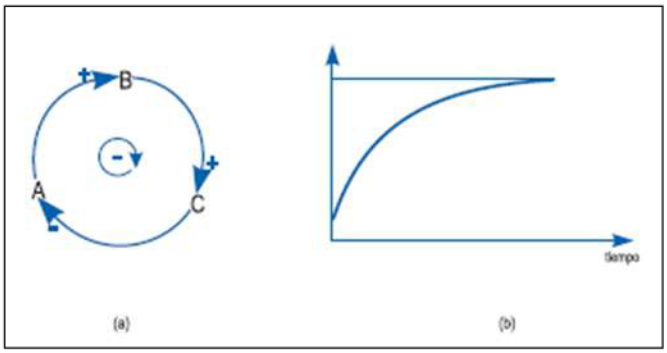


Figura 4 - Bucle Realimentación Negativo [11]

Características:

- Representa un tipo de situación que se da muy frecuentemente, en la que se trata de decidir acciones para modificar el comportamiento con el fin de alcanzar un determinado objetivo.
- Se denominan bucles reguladores o estabilizadores.
- Un bucle de retroalimentación negativa se forma por un número impar de influencias negativas.
- La resistencia al cambio tratando de mantener un objetivo es la principal manifestación de un bucle de retroalimentación negativa.
- Si no se detectan pueden generar comportamientos problemáticos.

1.2.3 Bucle Realimentación Positiva

En contraposición a los bucles de realimentación negativa, existen los Bucles de Realimentación Positiva. Esta estructura y su comportamiento se explicarán en presente punto.

Gráficamente un bucle de realimentación positiva se representa en la Figura 5(a). En ella se presenta la influencia de 3 variables A, B, C. Donde A influye positivamente sobre B, esta influye positivamente sobre C, y finalmente C influye también positivamente sobre A.

Analizando detalladamente el conjunto de influencias, en la primera de ellas, un incremento de la variable A hará que B se incremente, ya que existe una influencia positiva de la primera sobre la segunda variable. Igual situación ocurrirá con C, al incrementarse B. Para que finalmente el incremento de C, haga que A también se incremente.

Esta serie de influencias positivas entre todas las variables, hace que el sistema crezca indefinidamente, produciendo una situación de inestabilidad en el sistema.

Una manera sencilla de identificar a un Bucle de Realimentación Positiva, es a partir de que el número de influencias positiva sea impar.

El comportamiento en el tiempo de un bule de retroalimentación positiva se puede ver en la Figura 5(b).

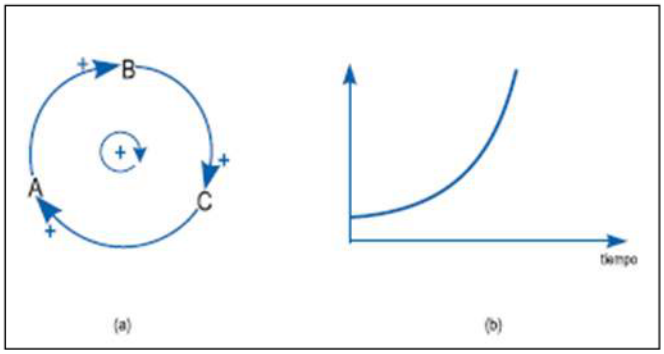


Figura 5 - Bucle Realimentación Positivo [11]

Características

- Se trata de un bucle en el que todas las influencias son positivas (o negativas en cantidades pares si las hubiere)
- Representa un proceso en el que un estado determina un acción que a su vez refuerza este estado y así indefinidamente.
- El crecimiento es explosivo, exponencial. El cambio se amplifica introduciendo más cambio.
- El efecto de este bucle se conoce más comúnmente como circulo vicioso o bola de nieve.
- Su efecto es contrario al Bucle de retroalimentación negativa, ya que este es desestabilizante.

1.2.4 Retrasos

En algunos casos interesa, además, distinguir entre influencias que se producen de forma más o menos instantánea e influencias que tardan un cierto tiempo en manifestarse. En este último caso, se tienen influencias a las que se asocian retrasos. En el diagrama de influencias, si A influye sobre B, y esta influencia tarda un cierto tiempo en manifestarse, entonces se añaden dos trazos sobre la flecha correspondiente. En la Figura 6(a) se muestra un bucle de realimentación negativa en el que la influencia entre C y A se produce con un retraso, por lo que la flecha correspondiente presenta dos trazos.

Los retrasos pueden tener una enorme influencia en el comportamiento de un sistema. En los bucles de realimentación positiva determinan que el crecimiento no se produzca de forma tan rápida como se esperaría y en los de realimentación negativa su efecto es más evidente, dado que su presencia puede determinar que ante la lentitud de los resultados se tomen decisiones equivocadas que conduzcan a inestabilidades en el sistema. Así en la Figura 6(b) se muestra el posible comportamiento del sistema de la

Figura 6(a), en el que se produce una oscilación en torno a la meta perseguida. Precisamente, el análisis de estas oscilaciones en un sistema con retrasos en la transmisión de información se encuentra en los orígenes de la dinámica de sistemas.

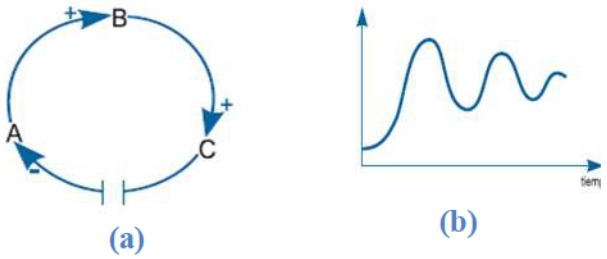


Figura 6 - Bucle Realimentación Negativo con Retraso [11]

1.3 Arquetipos Sistémicos

La palabra arquetipo viene del griego *Archetypos*, que significa "el mejor de su clase". Los arquetipos de sistemas fueron desarrollados en la empresa Innovation Associates a mediados de los años ochenta, en una época en la cual el estudio de la dinámica de sistemas dependía de diagramas causales complejos y del modelado por computadora basándose en ecuaciones matemáticas para el establecimiento de las relaciones existentes entre las variables.

En la mencionada empresa se buscaba la manera de comunicar los conceptos de una manera más simple y sencilla. Luego se desarrollaron una serie de diagramas que simplificarían las conductas más comunes de los sistemas, definiendo a los arquetipos como "*Herramientas accesibles que permiten construir hipótesis creíbles y coherentes acerca de las fuerzas que operan en los sistemas. Los arquetipos también constituyen un vehículo natural para clarificar y verificar modelos mentales acerca de esos sistemas*" [18].

Dentro de los arquetipos más comunes se encuentran

- Límites de Crecimiento Sigmoidal.
- Desplazamiento de Carga.
- Compensación Entre Proceso y Demora.

Estos arquetipos se describen a continuación.

1.3.1 Arquetipo Límite de Crecimiento Sigmoidal

Un proceso se pone en marcha para alcanzar un resultado deseado, en este proceso de crecimiento también generan efectos secundarios que pueden atentar contra el objetivo perseguido por el sistema.

Los arquetipos agrupados bajo este criterio representan aquellos sistemas que en algún momento de su vida encuentran un límite a su crecimiento. Presentan una estructura donde hay un proceso reforzador de crecimiento que opera por sí mismo durante un tiempo. Luego se encuentra con otro proceso compensador que tiende a estabilizarlo, y que opera para limitar el crecimiento. Este arquetipo puede observarse en la siguiente Figura 7.

El efecto del bucle positivo, pero este crecimiento tiene un límite al encontrarse con el bucle negativo, cuando el sistema alcanza un nivel de crecimiento considerable, el efecto se invierte, y el sistema dominante pasa a ser el negativo cuál hace que el crecimiento se detenga y luego disminuya.

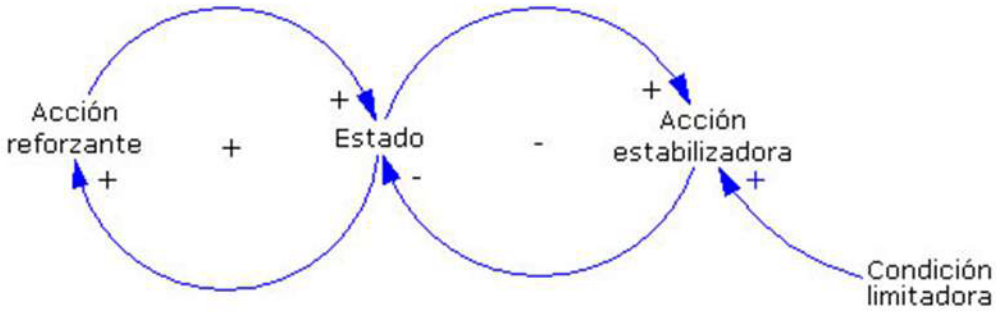


Figura 7 - Arquetipo Limite de Crecimiento Sigmoidal

1.3.2 Arquetipo Desplazamiento de Carga.

Este arquetipo se presenta en aquellos sistemas donde se usa una "solución" sintomática o de corto plazo para corregir un problema, y donde esta acción presenta resultados inmediatos aparentemente positivos, pero con la aparición de un efecto colateral

En la medida que esta corrección se usa reiteradamente, la solución fundamental es para atacar el síntoma del problema de fondo, estas se aplican cada vez menos. Con el tiempo, las aptitudes para la solución fundamental se atrofian, creando mayor dependencia respecto de la solución sintomática. En la Figura 8 se observa el arquetipo descripto.

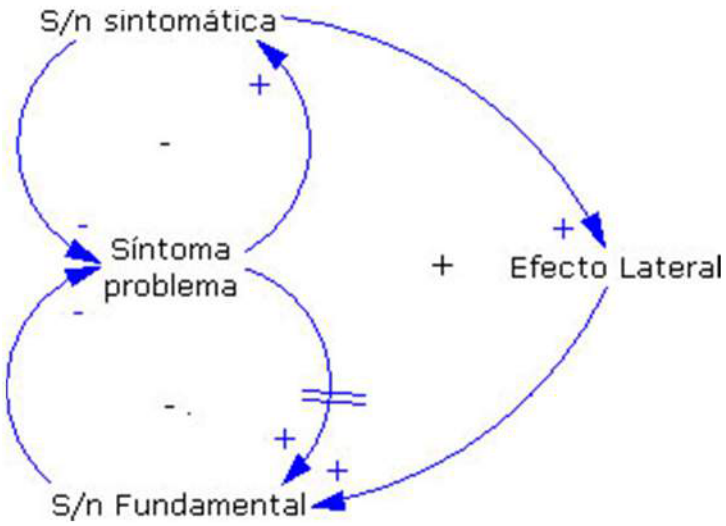


Figura 8 - Arquetipo Desplazamiento de Carga

1.3.3 Compensación entre Proceso y Demora

Este arquetipo presenta un comportamiento donde el sistema en funcionamiento actúa y desarrolla actividades con mira hacia una meta con condiciones reales, y donde se adapta el funcionamiento del sistema, por medio de una acción correctiva en respuesta a una realimentación demorada.

En caso de no ser conscientes de la demora al momento de realizar las acciones correctivas sobre el sistema, podrían llegar a realizarse más acciones de las necesarias o a veces desistir de las mismas por qué no se ve ningún progreso inmediato.

En la Figura 9 se aprecia el diagrama de influencia de este arquetipo.

Figura 9 - Arquetipo Compensación entre Proceso y Demora

1.4 Metodología dinámica de sistemas

Se ha visto en los apartados anteriores del presente capítulo, un tipo de lenguaje para la representación del comportamiento de los sistemas a partir de estructuras básicas y de la interacción de sus componentes. Este tipo de lenguaje sistémico elemental está conformado por bucles de realimentación. Si bien, este forma de representación es denominada dinámica de sistemas, en un sentido más estricto la dinámica de sistemas hace referencia a una metodología presentada por J. Forrester [19].

Forrester es un ingeniero que inició su carrera profesional trabajando en servomecanismos y en diseño de computadores, pudo comprender que un sistema con oscilaciones tendía a la meta perseguida a partir de que se producían retrasos en la transmisión de la información. En ese trabajo comparó la manera de trabajar de los servomecanismos e intuyó que eran similares, por lo tanto podían representarse de maneras similares, esta comparación la llevo adelante con el desarrollo de lo que en principio denominó dinámica industrial, y que luego se denominaría dinámica de sistemas.

1.4.1 Fases de la Metodología Dinámica de Sistemas

En el presente apartado se describen las distintas fases que componen la metodología, dichas fases son: Conceptualización, Formulación, y Análisis y Evaluación [11].

La primera fase se denomina **Fase de Conceptualización**. La cuál tiene como objetivo conocer en amplitud y profundidad el dominio del problema, es aquí donde se toma conocimiento acerca del sistema o fenómeno real, y la adopción de cierta perspectiva del mismo para luego definir con precisión los aspectos del problema y describirlos en forma clara, breve y precisa. Esto se logra mediante el relevamiento a través de material bibliográfico, charlas con seniors y con quienes tengan conocimiento del sistema a modelar.

De manera resumida podría decirse que básicamente la fase consta de tres sub-fases que pueden denominarse:

- Descripción verbal del sistema.
- Definición y alcance del sistema.

- Diagrama causal del sistema.

Esta segunda fase de la metodología se denomina **Fase de Formulación**. Con los diagramas construidos en la fase anterior, se procede a la construcción de otro modelo. Este modelo denominado diagrama de Forrester es un modelo matemático formal, donde se escriben las ecuaciones que son la representación de las distintas influencias y sus efectos modelados en los diagramas, y que luego permitirán transcribirlas a una herramienta informática para su posterior simulación.

Este otro modelo permite representar la dinámica del comportamiento del sistema. De manera resumida podría decirse que en esta etapa existen las siguientes sub-etapas:

- Construcción del diagrama de Forrester
- Determinación y desarrollo de las ecuaciones para la simulación,
- Programación de las ecuaciones.

Finalmente la tercera fase se denomina **Fase de Evaluación**. Consiste en el análisis del sistema, y se lo somete a distintas pruebas que permiten observar el comportamiento del mismo frente a diversos criterios de aceptabilidad. Estas pruebas y sus resultados permiten evaluar la validez y calidad del sistema, involucran la evaluación de la consistencia lógica de las hipótesis, la comprobación y comparación entre las trayectorias generadas por el modelo y las registradas en la realidad. Esta fase incluye el Análisis de sensibilidad explicado en la sección 1.4.3.

Esta fase al igual que las anteriores posee un conjunto de sub-fases que podrían denominarse:

- Análisis del modelo
- Evaluación
- Comunicación de resultados
- Implementación del modelo.

1.4.2 Utilizando la metodología de Dinámica de Sistemas

La metodología presenta una serie de elementos esenciales como ser Diagramas de Influencia, Diagrama de Forrester y los diferentes tipos de variables. De Manera general se detalla cómo utilizar esta metodología para construir modelos de simulación.

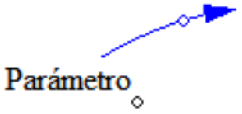
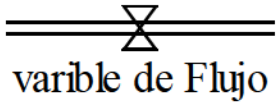
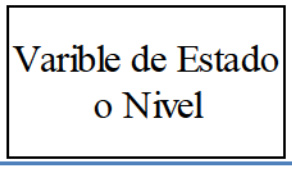
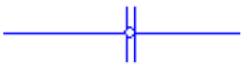
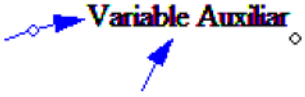
Con los datos e información recabados en la primera de las fase y que resulten oportunos para el dominio del problema a resolver, se procede a la primera aproximación del diseño del modelo mediante el desarrollo del Diagramas de Influencia. Con este tipo de diagramas se pueden identificar los arquetipos sistémicos que lo componen para así poder comprender las estructuras del sistema y las fuerzas que en el operan.

En la segunda fase a partir del Diagrama Causal se debe realizar proceso de transformación del mismo a al Diagramas de Forrester. En esta fase también es primordial identificar las variables, establecer su clasificación en “Niveles”, “Flujos” y “Auxiliares”, también es relevante realizar la correspondiente asignación de valores y unidades de medida a las distintas constantes y variables que permitirán posteriormente la simulación del sistema.

En principio Forrester distinguió tres grupos de variables: de estado, de flujo y auxiliares. Las variables de estado, representan las magnitudes cuya evolución es especialmente significativa. Las variables de flujo, están asociadas a las variables de estado y son las que determinan la variación de estas a lo largo del tiempo. Finalmente las variables auxiliares representan pasos intermedios para la determinación de las

variables de estado a partir de las variables de flujo asociadas. En la Tabla 1 se puede ver los elementos de un diagrama de Forrester y su representación en Vensim.

Tabla 1 – Elementos del Diagrama de Forrester de Vensim

Elementos del Diagrama de Forrester	Representación en Vensim
Canales de información: transmiten información que no permanece en el modelo, sino que simplemente se transmite entre las distintas variables.	
Canales Materiales: permiten la comunicación entre flujos y niveles. Se asume que a través de estos canales transmiten valores que representan magnitudes físicas.	
Constantes y Parámetros: almacenan valores que son utilizados en los cálculos por las variables auxiliares o en los flujos. De las constantes y parámetros siempre parten Canales o flujos de Información.	
Variables de Flujo: Representan funciones temporales. Relacionan a unas variables con otras y representan las relaciones causales que existen entre sí, expresando así, las acciones resultantes de las decisiones tomadas en el sistema que causan la variación del sistema.	
Variables de Nivel: corresponden a las variables de estados de la teoría de sistemas, y representan las variables cuya evolución es significativa para el estudio del sistema.	
Nube: representa una fuente, un pozo o sumidero. Puede interpretarse como un nivel que no tiene interés y es prácticamente inagotable.	
Retardos/Retrasos: se los utiliza para simular retrasos de tiempo en los canales de transmisión de flujo de materiales o de información.	
Variables Auxiliares: Son utilizadas para cálculos intermedios, y que son utilizados por otras variables. A Ellas siempre llegan Canales de Información.	
Shadow: este componente, es especial ya que se lo utiliza para referenciar en diferentes vistas una variable ya existente o una variable de sistema (Variable de control de simulación).	

Con el diagrama de Forrester finalizado, se procede a la programación de las fórmulas/ecuaciones mediante alguna herramienta computacional, en el presente trabajo se utiliza VenSim PLE 5.7ª (Versión Académica) [20].

Si bien, la metodología prevé los elementos descriptos en la Tabla 1, las variables que se detallan a continuación son específicas de la herramienta Vensim utilizada para la simulación del modelo correspondiente al presente trabajo.

- **Time**: Variable que representa el instante actual de la simulación en el tiempo.
- **INITIAL TIME**: constante que indica el instante en el cual se iniciará la simulación.
- **FINAL TIME**: constante que indica el instante en el cual finalizará la simulación.
- **LOOKUP**: Función discreta del tipo F(x,y), donde para cada valor de "x" se obtiene el correspondiente valor asociado "y". Para valores comprendidos entre dos "x" por ejemplo x1 y x2, utiliza un crecimiento lineal.

Además de las variables descriptas, VenSim posee un amplio conjunto de funciones disponibles. En el presente proyecto se utilizaron las siguientes:

- **ABS**({x}): retorna el valor absoluto del valor x.
- **DELAY FIXED**({in} , {dtime} , {init}): genera una demora fija de tiempo *dtime*, manteniendo constante el valor de *in* por ese periodo.
- **IF THEN ELSE**({cond} , {ontrue} , {onfalse}): función condicional que si el resultado lógico de evaluar *cond* es verdadero se ejecuta *ontrue*, en caso contrario *onfalse*
- **INTEGER**({x}): retorna el valor entero de *x* más cercano a 0(cero).
- **LOOKUP BACKWARD**({lookup} , {x}): permite controlar e interpolar el valor de entrada para *x* en la variable tipo *lookup*.
- **LOOKUP FORWARD**({lookup} , {x}): retorna el valor asociado a *x* de la variable *lookup*
- **PULSE TRAIN**({start} , {duration} , {repeattime} , {end}): genera una serie de pulsos que se inician en *start* y cuya duración es *duration*. Estos pulsos se repiten en el tiempo cada *repeattime* periodo hasta el instante temporal establecido en *end*.
- **QUANTUM**({x} , {base}): trunca *x* al entero múltiplo de *base* más cercano a cero.
- **RANDOM UNIFORM**({min} , {max} , {seed}): genera un número aleatorio comprendido entre *min* y *max*, dada una semilla establecida en *seed*.

Si bien, no existen reglas precisas de cómo hacer la trasformación de Diagramas de Influencias a Diagramas de Forrester, en [21] se propone una forma de abordar este proceso siguiendo 3 sencillos pasos:

1. Hacer una fotografía mental al sistema y lo que salga en ella (personas, km2, litros, tareas,..) eso son Niveles.
2. Buscar o crear unos elementos que sean "la variación de los Niveles", (personas/día, litros/hora,) y esos son los Flujos.
3. El resto de elementos son las Variables Auxiliares.

Estos pasos sirven como regla general, luego se deben hacer los retoques necesarios, en los cuales algunos de los niveles pueden ser constantes o variables en vez de definirlos como niveles.

1.4.3 Análisis de Sensibilidad

En todo modelo de dinámica de sistemas se involucran y vinculan variables de tipo cuantitativo con variables de tipo cualitativo, este vínculo entre variables tan diferentes genera inconvenientes que llevan al sistema a tener un cierto grado de imprecisión.

Cuando se realiza la prueba de un sistema en algunas oportunidades es importante analizar cómo se comportaría este si se modifican los valores, aunque sean próximos a los elegidos inicialmente, a ciertas variables, a un parámetro, o una forma funcional determinada a la expresión que relaciona dos variables. El comportamiento del sistema bajo estas condiciones de cambio es lo que aborda el Análisis de Sensibilidad.

"El análisis de sensibilidad consiste en un estudio sistemático de cómo afectan a las conclusiones de un modelo las posibles variaciones en los valores de los parámetros y en las relaciones funcionales que incluye" [11]

El proceso de análisis de sensibilidad consiste en seleccionar la variable del modelo que se quiere estudiar, modificar su valor, ya sea en porcentajes o con valores extremos, y ver como estos nuevos valores afectan al resto de las variables del modelo. Un inconveniente que se tiene al hacer este tipo de análisis es, que se deja de lado la posibilidad de que dos o más variables cambien de manera conjunta su valor, pero esto se puede reducir, recurriendo a formas más elaboradas de análisis de sensibilidad como son las que permite la aplicación del Método de Montecarlo.

De esta manera se puede determinar para que variables, la estructura del sistema resulta sensible. Se dice que el modelo es sensible a determinadas variables, si los cambios hechos a los valores de esta afectan al comportamiento y a las conclusiones de se puedan obtener.

El análisis de sensibilidad de un modelo constituye uno de los elementos esenciales para su evaluación. Por una parte permite ver la robustez del modelo y en qué medida este, es o no sensible a variaciones en su estructura: por otra parte, da respuesta a cuáles son los puntos más sensibles del modelo sobre los cuales en el proceso real serán más efectivas.

2 Gestión de Proyectos de Software: Metodologías Ágiles y Scrum

En el presente capitulo se desarrollará una introducción histórica de las metodologías de desarrollo y a continuación se presentará el concepto de metodologías ágiles, el manifiesto ágil, y también las metodologías presentes más usadas.

Seguidamente se verá en detalle SCRUM y sus elementos principales, Bloques de Tiempo, Artefactos y el ciclo de vida asociado.

2.1 Proyectos

En [22] se define un proyecto como, "Un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. La naturaleza temporal de los proyectos indica un principio y un final definidos"

En esta definición se pueden ver dos características fundamentales. La primera, es que todo proyecto es un esfuerzo para obtener un producto en un espacio finito de tiempo, de ahí su naturaleza temporal. La segunda, es que los proyectos tienen como resultado un producto o servicio único e irrepetible. Si bien diferentes proyectos pueden tener elementos comunes y presentar características similares el producto final es único.

Dicho de otra manera, aunque diferentes proyectos generen el mismo software, estos son únicos, ya que se ejecutarán en lugares diferentes, personas diferentes, bajo distintas circunstancias, etc. Es así que los productos finales de un proyecto son duraderos, perdurables en el tiempo, de allí que el producto debe ser considerado no solo pensando en el presente, sino que, previendo situaciones futuras.

2.1.1 Administración de proyectos

Para que el objetivo general y los objetivos específicos de un proyecto puedan ser alcanzados, es necesario que los recursos de la organización -como ser tiempo, dinero, recursos humanos e información entre otros- que están disponibles y que serán utilizados estén gestionados de la mejor manera posible.

La disciplina de Administración de Proyectos es la que se especializa en estudio de cómo lograr que proyecto se complete de la mejor manera. Formalmente se puede decir que esta consiste en "la planificación, organización, coordinación, liderazgo y control de recursos para alcanzar el objetivo del proyecto. El proceso de administración de proyectos involucra la planificación del trabajo y la puesta en marcha del plan" [23].

La administración o gestión de proyectos puede ser entendida como un proceso continuo, encuadrado en una estrategia general de la organización y apoyada por una serie de herramientas como por ejemplo PERT, CPM y GERT. Estas herramientas permiten y facilitan el control y seguimiento de las actividades del proyecto, tanto de las de gestión, relacionadas con la administración de la organización, construcción del producto, personas, como las de desarrollo, las cuales se centran en el desarrollo mismo

del proyecto, y que permiten construir un producto o servicio final. En este trabajo se detallaran particularmente los proyectos de desarrollo de software.

2.1.2 Fases de un proyecto

En la mayoría de los casos los proyectos que buscan el mismo objetivo tienen una serie de fases que son comunes. Esto puede parecer trivial pero desde la perspectiva de un proyecto, es primordial para el administrador saber en qué momento del proyecto se encuentra, ya que de esto dependerán las actividades deba realizar para la normal continuidad del proyecto.

La clasificación general del párrafo anterior puede detallarse más y alcanzar un nivel de granularidad tal que se puede dividir a un proyecto de software en fases más específicas y de esa manera llevarlo adelante de la mejor forma. Así, la carga total de trabajo se divide en componentes menores facilitando el desarrollo y seguimiento de las actividades.

En [22] se propone dividir un proyecto en las siguientes cinco fases genéricas:

- Iniciación
- Planificación
- Ejecución
- Seguimiento y Control
- Cierre

En el caso de los proyectos de software, existen metodologías que tienen fases más especializadas y específicas para el producto/objetivo que se desea obtener. En la Sección 2.2 se dará una introducción a distintas metodologías aplicadas al desarrollo de software, particularmente las metodologías ágiles y dentro de ellas Scrum.

2.1.3 Administrador de un Proyecto

La relación existente entre los diversos factores que están involucrados en un proyecto es tan fuerte y estrecha que si alguno de ellos cambia, es muy probable que al menos uno de los otros se vea afectado. Por ejemplo, un adelanto en el cronograma a menudo implica aumentar el presupuesto, a fin de añadir recursos adicionales para completar la misma cantidad de trabajo en menos tiempo.

Para poder conseguir la coordinación del equipo y la asignación de los recursos disponibles para alcanzar el objetivo propuesto en el proyecto surge el rol de Administrador de Proyectos, que es el encargado de llevar adelante las acciones de coordinación y asignación.

Administrar un proyecto por lo general implica un conjunto de acciones tales como:

- Identificar requisitos
- Abordar las diversas necesidades, inquietudes y expectativas de los interesados
- Según se planifica y efectúa el proyecto, equilibrar las restricciones contrapuestas del proyecto que se relacionan.

Según la IPMA [24] el administrador del proyecto o la persona a cargo del mismo, debe manejar o tener competencias suficientes en 3 aspectos:

- **Competencia técnica**: es aquella que le permite conocer la tecnología principal del proyecto y así conocer los aspectos claves del mismo, y a partir de esto poder planificar el uso de recursos, presentar soluciones y propuestas que mejoren la calidad del producto final.
- **Competencia de comportamiento:** debe poseer una capacidad destacada para las relaciones interpersonales, autoridad personal, capacidad de convicción. Esta competencia es más bien de tipo político, es válido recordar que el administrador es el nexo principal entre el dueño del sistema y el equipo de trabajo, además, es el mediador entre los miembros del equipo, y es la referencia máxima de estos.
- **Competencia Contextual:** la capacidad de conocer el contexto organizacional de un proyecto, y la capacidad para funcionar en una organización por proyectos, donde el rol que desempeña deriva de las actividades, responsabilidades y requisitos típicos utilizados en la práctica.

Dentro de las actividades que un administrador de proyecto desarrolla están:

- Dirigir, supervisar y coordinar la realización y desarrollo de los trabajos.
- Hacer cumplir los objetivos estipulados, dentro de los plazos y costos preestablecidos.
- Hacer cumplir las normas de funcionamiento y las condiciones estipuladas en la planificación inicial del proyecto.
- Asegurar el nivel de calidad técnica de los trabajos.
- Coordinar las actividades entre usuarios y técnicos involucrados en el proyecto.

2.1.4 Gestión de Riesgos

Una tarea importante del Administrador de Proyectos es anticiparse a los riesgos que podrían afectar el normal desarrollo del proyecto o a la calidad del software producido, poder desarrollar y emprender acciones para evitar que estos riesgos influyan de sobremanera al proyecto resulta una tarea esencial en la gestión.

Según Ian Somerville [25], los riesgos podrían clasificarse en tres grupos:

- Riesgos del proyecto. Estos afectan la calendarización o los recursos del proyecto. Por ejemplo la pérdida de un diseñador experimentado.
- Riesgos del producto. Estos afectan a la calidad o al rendimiento del software que se está desarrollando. Por ejemplo, que el rendimiento en un componente comprado sea menor que el esperado o estimado.
- Riesgos del negocio. Estos afectan a la organización que desarrolla o suministra el software. Por ejemplo, la introducción de un nuevo producto por parte de un competidor del mercado.

2.1.5 Ley de Brooks

Dentro de los riesgos más importantes en un proyecto, en "The Mythical Man-month" [26], se enuncia que "*Agregar mano de obra a un proyecto de software atrasado lo atrasa aún más*". Este enunciado luego se conoce hoy día como Ley de Brooks.

En el presente modelo esta ley es concebida en función de las siguientes suposiciones:

- El nuevo personal requiere de cierto tiempo de entrenamiento y conocimiento de la metodología.
- Más personal en el proyecto trae más sobrecarga en la comunicación.
- El personal experimentado tiene una influencia mayor en la producción que uno nuevo.

El modelo diseñado en este trabajo no pretende ser una "Bala de Plata", pero si ser de ayuda para los encargados de la administración de proyectos que utilicen Scrum como metodología de gestión para que puedan actuar de manera proactiva ante situaciones de riesgo en los proyectos. Cabe destacar que la Ley de Brooks está representada en el modelo construido.

2.2 Introducción a las metodologías de desarrollo de software.

Inicialmente los proyectos que se llevaban a cabo eran de tipo artesanal desarrollando las actividades de manera desorganizada y caótica, si bien existía algún tipo de organización y coordinación la misma no implementaba ninguna metodología, todo consistía en programar, realizar correcciones y continuar programando hasta lograr el objetivo [25].

Esta manera de trabajo podía funcionar en tanto y en cuanto el proyecto para desarrollar el sistema fuera relativamente pequeño, el problema se presentaba al momento de que los sistemas comenzaban a crecer por la aparición de nuevos requerimientos de los usuarios.

La necesidad de formalizar el desarrollo de proyectos surgió en el ámbito militar en los años 1950 donde el desarrollo de complejos sistemas militares requería de un nivel de coordinación elevado, y donde el trabajo conjunto de equipos y disciplinas diferentes, eran esenciales para la construcción de sistemas únicos.

Durante los años 1960 surgieron las denominadas metodologías clásicas, a partir de la aparición de organizaciones que permitieron el desarrollo del cuerpo del conocimiento para la gestión de proyectos y así ofrecer cierto nivel de calidad al resultado. Este tipo de metodologías permitía una gestión conocida como gestión de proyectos predictiva, y la planificación se basa en un análisis detallado del trabajo a realizar y su respectiva descomposición en tareas, su principal objetivo es conseguir que el producto final se obtenga según lo "previsto"; y basaba el éxito del proyecto en tres puntos principales: agendas, costos y calidad.

Las metodologías utilizadas a mediados del siglo XX permitían generar productos que se mantenían en el mercado por años, ya que las necesidades y cambios en los mismos no eran exigentes o no sufrían variaciones a lo largo del tiempo. Luego de los avances tecnológicos y los negocios a escala global dieron lugar a la necesidad de un rápido desarrollo de productos de software en algunos segmentos del mercado y pese a ser una necesidad de las empresas reducir el time-to-marked. Es así como a finales del XX surgen las denominadas metodologías ágiles que a diferencia de las clásicas son mucho menos burocráticas y son mejores ante requerimientos cambiantes a lo largo del desarrollo del proyecto.

2.3 Metodologías ágiles

En 2001, un grupo de críticos fueron convocados y reunidos por Kent Beck [27] (quien un tiempo antes había expuesto una nueva metodología denominada "Extreme

Programming") a fin de realizar una mejora a los modelos para el desarrollo de software.

De la reunión surgen dos grandes conceptos, uno es el término "Métodos Ágiles" a lo que aparecía como una alternativa a los modelos formales existentes en ese momento, (CMM-SW, SPICE, Etc). Se buscaba así una alternativa a los procesos de desarrollo tradicionales caracterizados por su rigidez y muy dirigidos a la documentación que se genera tras cada una de las actividades desarrolladas, y el otro concepto que se reflejó en el resumen de esa reunión al que se llamó "Manifiesto Ágil" [28], Dicho manifiesto se presentó en 4 postulados:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

El objetivo del documento final fue delinear y presentar una serie de valores y principios que servirían para comprender la filosofía de trabajo, el objetivo y la finalidad de las metodologías ágiles; se buscó que estos valores y principios permitieran a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Los principios del Manifiesto Ágil se listan a continuación:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Cabe mencionar que las metodologías más relevantes que en la actualidad comparten el calificativo de "Ágiles" son las siguientes:

- XP (Extreme Programming). [27] Centrada en maximizar las relaciones interpersonales como clave para el éxito, promueve el trabajo en equipo, atendiendo el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, manteniendo una comunicación fluida entre todos los participantes, la simplicidad en las soluciones y la fuerza para enfrentar los cambios. Es principalmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

- Crystal Methodologies [29] [30] Es un conjunto de metodologías propuestas por Alistair Cockburn para el desarrollo de software. Están centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Se considera al desarrollo de software como un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. De acurdo con el tamaño del equipo dependen las políticas a adoptar, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

- ASD (Adaptative Software Development) [31]. Propuesta por Jim Highsmith, tiene como principales características las siguientes: es iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y por último en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar proceso.

- FDD (Feature Driven Development) [32]. Peter Coad y Jeff De Luca proponen un proceso iterativo que consta de una serie de pasos, con iteraciones cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema a partir de una lista de características que debe reunir el software.

- DSDM (Dynamic System Development Method) [33]. Especifica el marco para desarrollar un proceso de producción de software. Se crea con el objetivo de crear una metodología RAD (rapid aplication development) unificada. Sus características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Consta de cinco fases: estudio viabilidad; estudio del negocio; modelado funcional; diseño y construcción; y finalmente; implementación. Las tres últimas son iterativas y existe realimentación a todas las fases.
- SCRUM [1]. Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos años. Está pensada para proyectos con un rápido cambio de requisitos. Como es la metodología en la cual se basa este trabajo, la misma será explicada detalladamente en la siguiente sección.

2.4 Scrum

Los constantes e impredecibles cambios de contexto generados por la evolución y la aparición de nuevas necesidades y tecnologías, y en consecuencia de los requerimientos de los usuarios han llevado a que las metodologías de desarrollo de software hayan tenido que adaptarse a estas necesidades.

Es por ello que las organizaciones para poder mantenerse en el actual mercado de manera competitiva necesitan que sus equipos puedan desarrollar software rápidamente y responder a los cambios que pueden surgir a lo largo del proyecto.

Para poder afrontar estos cambios, ya durante las décadas de 1970 y 1980, Nonaka y Takeuchi estudiaron la forma de trabajar de algunas empresas como Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M, Xerox, Hewlett-Packard, entre otras, pudieron observar que estas mantenían ventajas en relación a sus competidores en cuanto a la rapidez e innovación, ya que compartían pautas similares de trabajo, que eran totalmente diferentes al modelo clásico secuencial de la gestión de proyectos.

En estas empresas el trabajo no se realiza dividiéndolo entre equipos y especialistas diferentes. El producto emerge de la interacción constante de un equipo de élite, multidisciplinar que trabaja conjuntamente desde el principio hasta el final. Nonaka y Takeuchi compararon la forma de trabajar de estos equipos únicos y multidisciplinares, con los equipos de rugby cuando realizan el "Scrum", y el ambiente y entorno de trabajo que les proporcionaba la empresa lo llamaron "Campo de Scrum"

Desde el punto de vista metodológico Scrum es un marco de trabajo donde se aplican un conjunto de prácticas y herramientas para trabajar en equipos autodirigidos cooperativos con el fin de obtener el mejor resultado de un proyecto. La selección de éstas prácticas tiene su origen en un estudio realizado por Hirotaka Takeuchi e Ikujio Nonaka sobre "nuevas prácticas de producción" a mediados de los 80 en su artículo titulado "The new product development game" [34].

Fue aplicado por primera vez por Jeff Sutherland y Ken Schwaber a principios de los 90, quienes lo documentaron en el libro "Agile Software Development with Scrum [1]".

Scrum se compone de un conjunto de elementos como, Equipos Scrum y sus roles asociados; Bloques de Tiempo, Artefactos, y Reglas.

Dentro de los elementos relacionados a roles aparecen el ProductOwner, Scrum Master y el Team.

Los elementos de Scrum basados en bloques de tiempo son: Scrum son: la Reunión de Planificación de la Entrega, la Reunión de Planificación del Sprint, Scrum Diario, Sprint, la Revisión del Sprint, la Retrospectiva del Sprint

Scrum emplea cuatro componentes principales en su proceso. El Product Backlog, Sprint Backlog, Burndown de Versión, Sprint Burndown y Sprint Backlog en el transcurso de un Sprint. Estos se pueden apreciar en la Figura 10.

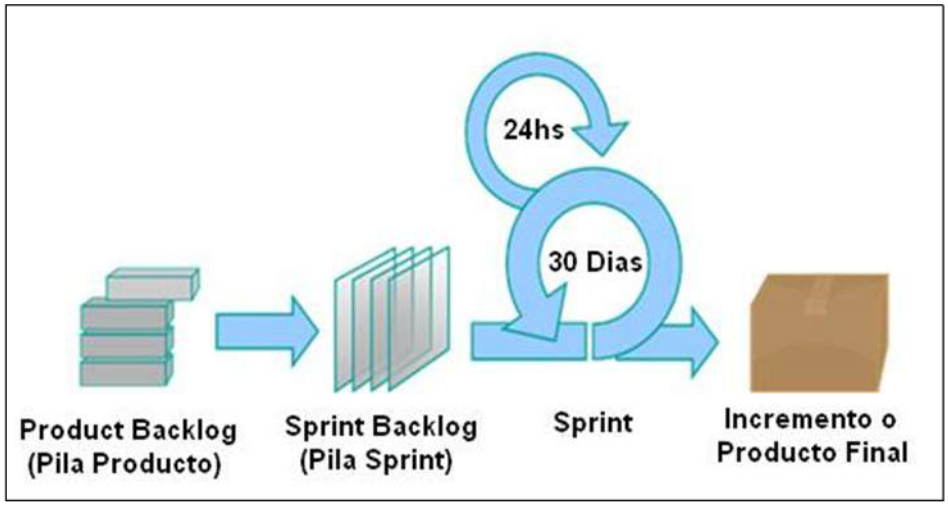


Figura 10 - Artefactos Scrum

A continuación se detallaran los elementos que componen un proyecto desarrollado siguiendo Scrum.

2.4.1 Roles

Scrum utiliza el concepto de Rol para referenciar al papel que cumple cada uno de los integrantes del proyecto durante su desarrollo, además permite tener trabajadores multifuncionales en lugar de simplemente desarrollar actividades según el nombre del puesto de trabajo tal como "tester", "desarrollador" o cualquier otro título que pudiera asignarse a un integrante.

Dicho de otra manera, los miembros del equipo acuden a donde está el trabajo y participan en todo lo que sea posible, por ejemplo si hay muchas tareas de pruebas, todo el equipo puede ayudar a realizar estas pruebas. Esto no significa que todos son generalistas; por supuesto que hay especialistas y cada integrante tiene una habilidad más fuertemente marcada, pero en Scrum los miembros del equipo trabajan en forma conjunta y aprenden nuevas habilidades de los demás.

Los roles que propone Scrum son los siguientes:

- **Responsable-Facilitador (Scrum Master):** Se trata de un rol que vela por la integridad de la metodología. Su principal objetivo es que el marco del trabajo se lleve a la práctica como se debe, que el clima del equipo sea el adecuado y que no haya intervenciones externas que perturben al equipo. Será el portavoz del equipo, atendiendo todo aquello que venga de afuera y transmitiendo todo aquello que venga de adentro. Quien realice este rol puede realizar también un rol dentro Equipo (Team), pero nunca podrá ser ProductOwner a la vez que Scrum Master. El Scrum Master es que coordina las actividades de planificación y es uno de los posibles usuarios del modelo de Simulación propuesto en este trabajo.
- **Propietario del Producto (Product Owner):** Es el responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo. Su principal objetivo será el de analizar el proyecto desde un punto de vista de negocio, ya que es el responsable del éxito del producto. El Dueño de Producto es la única persona responsable de gestionar el Product Backlog. Quien realice este rol puede realizar también un rol del tipo Team, pero nunca podrá ser Scrum Master a la vez que ProductOwner
- **Equipo (Team):** Son los encargados de desarrollar las funcionalidades del software. La ventaja que ofrece Scrum sobre otras metodologías es que se le da

libertad de gestión al Team, de forma que son ellos quienes crean el Sprint Backlog eligiendo qué hacer durante el siguiente Sprint. Sin embargo su creación se hace con la supervisión del ProductOwner. El conjunto de roles Team está formado por 7±2 personas conformando un equipo multidisciplinario, ya que en cada iteración se pasa por todas las fases de desarrollo de un proyecto. La composición del Team será la misma mientras dure el Sprint y no se harán nuevas modificaciones hasta que empiece uno de nuevo.

2.4.2 Bloques de tiempo

Los Bloques de Tiempo en Scrum son: la Reunión de Planificación de la Entrega, la Reunión de Planificación del Sprint, Scrum Diario, Sprint, la Revisión del Sprint, la Retrospectiva del Sprint. A continuación se explican los bloques.

- **Planificación del Sprint (Sprint Planning)**

El propósito de la planificación de la entrega es establecer un plan y unas metas que los Equipos Scrum y el resto de las organizaciones puedan entender y comunicar. Se divide en dos partes:

La primera fase tiene como objetivo establecer que ítems del Product Backlog List van a ser realizados durante el Sprint. En esta reunión el Dueño de Producto y Equipo (con la facilitación del Scrum Master) revisan los elementos de alta prioridad de la Pila de Producto en los que el dueño presenta mayor interés y que deben ser implementados para este Sprint.

En la segunda se decide cómo se van a alcanzar los objetivos del Sprint. En esta fase se crea la Sprint Backlog, indicando qué tareas debe desempeñar el equipo para cumplir con dichos objetivos. Se centra en la planificación detallada de tareas para saber cómo implementar los elementos que el equipo decide hacer. Es el Equipo quien selecciona los elementos de la Pila de Producto a las que se comprometen que estará al final del Sprint, comenzando por la parte de arriba de la Pila de Producto.

- **Reunión Diaria de Sincronización Del Equipo (Scrum Daily Meeting)**

El objetivo de esta reunión es facilitar la transferencia de información y la colaboración. Se realizan siempre en el mismo lugar, no duran más de 15 minutos.

- **Ejecución de la iteración (Sprint)**

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos. Suelen ser iteraciones de 1 a 4 semanas dependiendo de lo que se determine en el Team. El carácter de iterativo hace que todo nuevo Sprint comience inmediatamente después de finalizado el Sprint previo, y donde al final de cada iteración se debe proporcionar un resultado completo, un incremento del producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su costo y quedan repartidos en iteraciones y entregas.

Si bien de manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la replanificación de objetivos que se realiza al inicio de cada iteración, no es posible introducir nuevos requerimientos o

cambios o realizar modificaciones a los ya existentes durante el desarrollo del sprint. Es decir que una vez que el Equipo se compromete, cualquier cambio o adición debe esperar al siguiente Sprint.

De esta manera un Sprint se puede resumir como un bloque de tiempo iterativo en el cual se desarrolla o mejora una funcionalidad de un sistema para producir nuevos incrementos. Durante el mismo un producto puede ser diseñado, codificado y probado, evolucionando su arquitectura y diseño durante el desarrollo.

- **Demostración de Los Requisitos Completados (Sprint Review)**

El Sprint review proporciona un punto de inspección para el progreso del proyecto al final de cada Sprint. Basándose en esta inspección, se pueden hacer adaptaciones al proyecto.

Cuando finaliza el Sprint el equipo presenta el incremento del producto. La presentación del incremento tiene como objetivo facilitar la retroalimentación de la información y fomentar la colaboración. No puede llevar más de 2 horas.

- **Retrospectiva (Sprint Retrospective)**

Este bloque de tiempo tiene como propósito principal el de inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas, además en él se deben identificar y ordenar los elementos más importantes y las posibles mejoras a incluir.

Esta reunión se realiza después de la Sprint Review para poder incorporar el feedback y cumplimiento de expectativas. Se restringe la reunión a un bloque de tiempo de tres horas para Sprints de un mes. Para Sprints más cortos se reserva un tiempo proporcionalmente menor.

2.4.3 Artefactos principales de Scrum

Los Artefactos de Scrum incluyen: Scrum Taskboard, el Product Backlog, el Burndown de entrega, el Sprint Backlog, y el Sprint Burndown. Las características de los mismos se describen a continuación:

- **El Tablero de Tareas (Scrum Taskboard)**

La lista de objetivos a completar en la iteración se puede gestionar mediante una tabla o pizarrón de tareas (Scrum Taskboard). A lado de cada objetivo se ponen las tareas necesarias para completarlo, en forma de post-its, y se van moviendo hacia la derecha para cambiarlas de estado (pendientes de iniciar, en progreso, hechas). En la Figura 11 se puede ver un Ejemplo de Scrum Taskboad

- **Lista de Objetivos o Lista de Requisitos Priorizada (Product Backlog)**

Es una lista que contiene los objetivos/requerimientos priorizados que representan la visión y expectativas del Product Owner sobre el producto o proyecto. Es la única fuente de requerimientos para cualquier cambio a realizarse en el producto.

Es una lista dinámica, en constante evolución con el entorno para identificar lo que el producto necesita para ser adecuado, competitivo y útil. En ella se listan todas las características, funcionalidades, requerimientos, mejoras y correcciones que constituyen

cambios a ser hechos sobre el producto para entregas futuras. Los elementos del Product Backlog tienen como atributos la descripción, la ordenación y la estimación.

- **Lista de Tareas de la Iteración (Sprint Backlog)**

El Sprint Backlog es un conjunto de elementos de la Product Backlog seleccionados para el Sprint más un plan para entregar el incremento del producto y conseguir el objetivo del Sprint. El Sprint Backlog es una imagen visible y en tiempo real del trabajo llevado a cabo por el Team en el Sprint.

Esta lista permite ver las tareas donde el equipo está teniendo problemas y no avanza, con lo que le permite tomar decisiones al respecto. Es importante destacar que es el Team quien se organiza para alcanzar el objetivo, pudiendo agregar nuevas tareas o remover tareas innecesarias en cualquier momento si lo considera necesario para cumplir el objetivo propuesto, y debiendo actualizar las estimaciones cada vez que aparece nueva información. De manera resumida el Sprint Backlog solo puede ser modificado por el equipo.

Para cada uno de los objetivos/requisitos se muestran sus tareas, el esfuerzo pendiente para finalizarlas y la autoasignación que han hecho los miembros del equipo.

En Scrum se planifica y mide el esfuerzo restante necesario para desarrollar el producto. Esta gráfica suele utilizarse en lugar de un diagrama de PERT debido a que el camino crítico en un desarrollo ágil cambia diariamente, lo que haría obsoleto el diagrama de PERT cada día. Es por esto que no es útil una herramienta que modele el camino crítico a partir de actividades.

La solución a esto es utilizar alguna técnica que permita medir la velocidad de desarrollo de las actividades en la medida que el proyecto avanza. Para esto se utiliza el criterio del equipo a partir del cual se calcula diariamente el camino crítico, y se puede así recalcular el plan y la velocidad con la que se realiza el trabajo. En función de todo esto el equipo puede trabajar para acelerar o desacelerar la velocidad de desarrollo del trabajo para cumplir con la fecha de entrega.

- **BurdownChart**

El BurdownChart es un tablero que permite realizar diversas simulaciones: ver cómo se aplazan las fechas de entrega si se le añaden requisitos, ver cómo se avanzan si se le quitan requisitos o se añade otro equipo, etc.

En la Figura 11 en se presenta una modelo de Scrum Taskboard, y en ella se pueden apreciar el BurdownChart, el Sprint Backlog, y algunas tareas no planificadas comunes en Scrum.

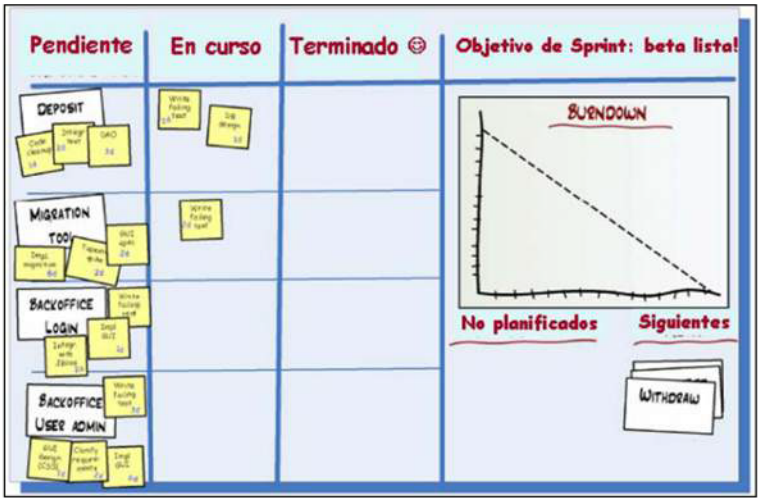


Figura 11 - Modelo Scrum Taskboard

2.4.4 Ciclo de vida

En Scrum el ciclo de vida se divide en 3 fases denominadas: Pregame, Development Game, Postgame, y cuyo vínculo se observan en la Figura 12.

- **Pregame Phase**

La fase de Pregame incluye dos subfases: Planning y Architecture

- **Planning:** Consiste en la definición del sistema que será construido. Para esto se crea la lista Product Backlog a partir del conocimiento que actualmente se tiene del sistema. En ella se expresan los requerimientos priorizados y a partir de ella se estima el esfuerzo requerido. El Product Backlog es actualizado constantemente con ítems nuevos y más detallados, con estimaciones más precisas y cambios en la prioridad de los ítems.
- **Architecture / High level Design:** El diseño de alto nivel del sistema se planifica a partir de los elementos existentes en el Product Backlog List. En caso de que el producto a construir sea una mejora a un sistema ya existente, se identifican los cambios necesarios para implementar los elementos que aparecen en la lista Product Backlog y el impacto que pueden tener estos cambios. Se sostiene una Design Review Meeting para examinar los objetivos de la implementación y tomar decisiones a partir de la revisión. Se preparan planes preliminares sobre el contenido de cada release.

- **Development o Game Phase**

Es la parte ágil de Scrum: En esta fase se espera que ocurran cosas impredecibles. Para evitar el caos Scrum define prácticas para observar y controlar las variables técnicas y del entorno, así también como la metodología de desarrollo que hayan sido identificadas y puedan cambiar. Este control se realiza durante los Sprints. Dentro de variables de entorno encontramos: tiempo, calidad, requerimientos, recursos, tecnologías y herramientas de implementación. En lugar de tenerlas en consideración al

comienzo del desarrollo, Scrum propone controlarlas constantemente para poder adaptarse a los cambios en forma flexible.

- **Postgame Phase**

Contiene el cierre del Release. Para ingresar a esta fase se debe llegar a un acuerdo respecto a las variables del entorno por ejemplo que los requerimientos fueron completados. El sistema está listo para ser liberado y es en esta etapa en la que se realiza integración, pruebas del sistema y documentación.

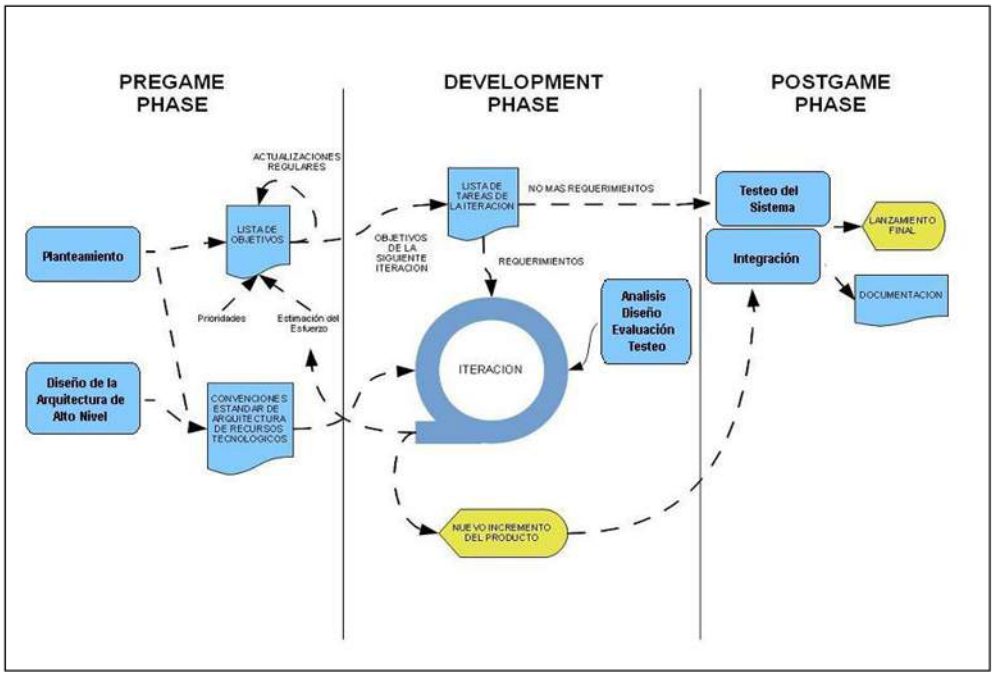


Figura 12 - Fases Scrum [35]

3

A continuación se detallan una serie trabajos donde se presentan proyectos y casos que si bien están desarrollados bajo otras metodologías ágiles como ser XP, APD, CFMethod, sirven como referencia y para presentar el presente apartado.

3.1 Modelo de Abdel-Hamid y Madnick

Fue el primer Modelo Dinámico de simulación para proyectos, y contiene las tres funciones principales de la dinámica de proyectos de desarrollo de software: la dirección, la producción y recursos humanos. El modelo mantiene los requisitos del proyecto estables a lo largo del ciclo de vida y se limitaba a los proyectos de tamaño medio. Simulaba proyectos de Desarrollo de software que se desarrollaban bajo la metodología tradicional común al momento que surgiera el modelo, por lo tanto no contaba con las fases de Definición de requisitos, Operación y mantenimiento, centrándose en Diseño, Codificación, Revisión, Corrección y Pruebas. En la Figura 13 se puede ver el modelo reducido del trabajo [36].

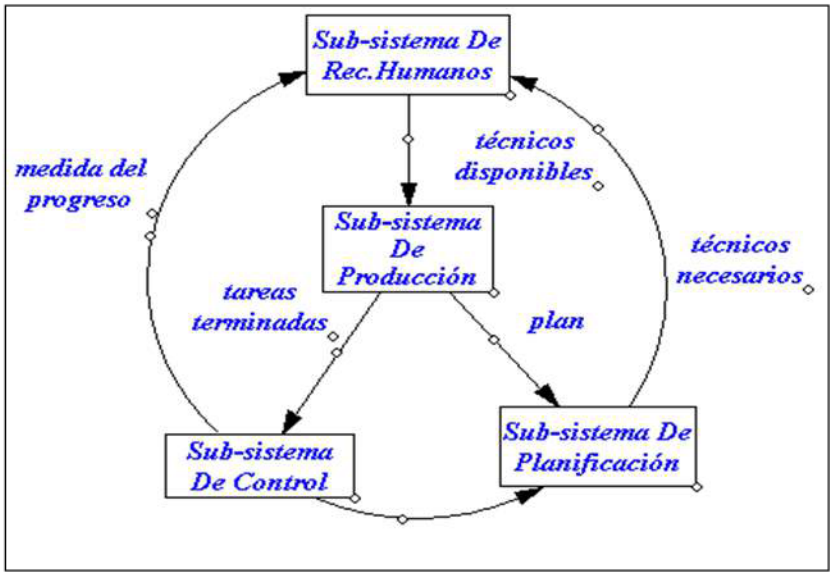


Figura 13 - Modelo Reducido de Abdel-Hamid y Madnick

3.2 Modelo Dinámico de Simulación de Proyectos de Software con XP

El objetivo general de este trabajo es el análisis del efecto que tiene el uso de la metodología XP sobre la gestión de proyectos, su representación y modelado mediante un sistema Dinámico. Como aporte principal presenta una herramienta de soporte a los

gerentes de proyectos que les permite trabajar y simular con diferentes situaciones de un proyecto que se desarrolla bajo metodología XP [5] [4].

A través de la utilización de VenSim como herramienta de simulación, la presentación de diversos Subsistemas mediante diagramas de Forrester, la aplicación de diferentes políticas de gestión expone diversos escenarios que facilitan un rápido y sencillo análisis del efecto sobre las variables involucradas en la administración de un proyecto de software.

Para la validación del modelo desarrollado se tomaron 3 (tres) casos reales, los cuales fueron analizados y utilizados como referencia para las corridas de las situaciones de práctica planificadas, concluyendo que el modelo desarrollado y los resultados obtenidos concuerdan con los casos provistos como modelos.

3.3 Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics

Éste trabajo presenta un modelo dinámico de sistemas para proyectos de desarrollo de software con metodologías ágiles denominado Agile Project Dynamics (APD). Éste modelo se basa en el estudio de varías metodologías ágiles, donde se identifican siete características compartidas denominadas: Agile Genome. El modelo capta cada uno de los siete genes como un componente importante del modelo [6].

- Historia: dividir el proyecto en partes manejables de funcionalidad, a veces llamados “rasgos”, “cuentos”, “casos de uso”, o “hilos”.
- Iterativo Incremental: el desarrollo se realiza en ciclos repetitivos (iterativo) e incremental.
- Refactorización: refinamiento del diseño y arquitectura del software para mejorar su mantención y flexibilidad.
- Micro Optimización: los equipos tienen las facultades para modificar aspectos del proceso o adaptarse dinámicamente a las circunstancias cambiantes. Pequeñas mejoras y cambios variables se hacen frecuentemente cuando sea necesario.
- Participación del Cliente: Cliente/Usuario participa en las demostraciones de la funcionalidad para verificar/validar características del proyecto. Con gran frecuencia se requiere la retroalimentación y clarificación de la incertidumbre. Disponibilidad para participar en las reuniones de desarrollo.
- Equipos Dinámicos: Factores "blandos" en relación con el equipo del proyecto. Las reuniones diarias, espacios de trabajo ágiles, programación en parejas, horario / la presión de grupo, el aumento de la experiencia, etc.
- Integración Continua: Las políticas y prácticas relacionadas con la gestión de la configuración, y crear y probar la automatización.

El modelo APD pretende ayudar en la comprensión del impacto que tienen las distintas combinaciones de prácticas ágiles, combinadas con diferentes políticas de gestión, sobre el desempeño del proyecto, en comparación con un enfoque en cascada. No se enfoca en predecir o volver a crear resultados de proyectos reales, sino para comparar el comportamiento de un proyecto bajo diferentes escenarios.

3.4 Dynamics of Agile Software Development

El trabajo presenta un estudio comparativo del desarrollo de un proyecto mediante una metodología ágil sin especificación de cuál, y a través de variados casos hipotéticos se analiza la relación entre el número de errores que pudieran existir durante el

desarrollo y la presión existente respecto al plazo para la finalización de un proyecto. [2].

La hipótesis que plantea es que niveles moderados de agilidad son más eficientes para obtener mejores resultados en los proyectos. A través de la utilización de Vensim como herramienta de simulación, el correspondiente diagrama de Forrester y diferentes políticas de gestión presenta escenarios donde el número de iteraciones y la duración de los mismos varían.

Concluye que la hipótesis planteada es verdadera y que el rendimiento del modelo es mejor cuando las longitudes de las iteraciones son de 50 jornadas de trabajo, en comparación a iteraciones de 20 días de trabajo que es lo más utilizado en la práctica real.

La utilización de casos hipotéticos para las pruebas y la falta de comparación de estos contra casos reales hace que los resultados obtenidos solo sean válidos para los escenarios planteados y no permitan analizar resultados que se adaptan a situaciones reales a partir de una validación concreta y más real.

3.5 Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment

El trabajo presenta un modelo para analizar el impacto de la presión del calendario sobre la eficacia económica en el proceso de mantenimiento de un proyecto mediante una metodología ágil llamada Critical Feature Method (CFMethod) [3].

Basándose en un diagrama causal, que los autores desarrollaron previamente y teniendo en cuenta la teoría analítica de la inversión del proyecto, analizan y simulan como la presión del calendario y la necesidad de un cambio rápido en el sistema influye en la economía y el costo de mantenimiento del sistema. Su estudio se basa en que en la medida que la presión del calendario se incrementa, la incertidumbre del costo del proyecto también se incrementa, o caso contrario en la medida que la presión del calendario disminuye, la incertidumbre del costo del proyecto también disminuye.

El estudio concluye que la eficacia del mantenimiento es alta cuando la presión horaria es baja, y, baja cuando la presión es alta, y a medida que aumenta la presión es probable que haya sucesos que podrían conducir en errores y reprogramación de actividades o tareas, por lo tanto retrasos y sobrecostos.

El trabajo, centra su estudio en una sola variable de las muchas que pueden influir en la administración de un proyecto, esta posibilidad de estudiar en concreto a una de las variables permite conocer de manera mucho más precisa como sería la administración del proyecto si esa variable fuera el factor determinante para el éxito o fracaso. Se genera así una visión parcial y concreta, pero no permite ver como otras variables influirían a lo largo de la administración de un proyecto de desarrollo de software.

3.6 Modelo Dinámico Reducido

Este modelo permite realizar estimaciones en las primeras etapas de un proyecto software donde aún se dispone de poca información sobre el mismo. Es un modelo aplicable especialmente en organizaciones donde existen pocos proyectos o bien en aquellas donde la información requerida para poder definir las características del entorno de trabajo del proyecto son escasas. También es aplicable a organizaciones o empresas que trabajan en entornos de desarrollos variables [37].

El modelo se ha obtenido a partir del estudio realizado a diferentes modelos dinámicos para proyectos de I+D y a modelos dinámicos para proyectos software, manteniendo la estructura de subsistemas propuestos por el Modelo de Abdel-Hamid y Madnick [36]. Además en él se recogen los principales bucles de realimentación y las

variables fundamentales de cualquier proyecto de software dinámico. En la Figura 14 se puede ver el diagrama causal del Modelo Dinámico Reducido.

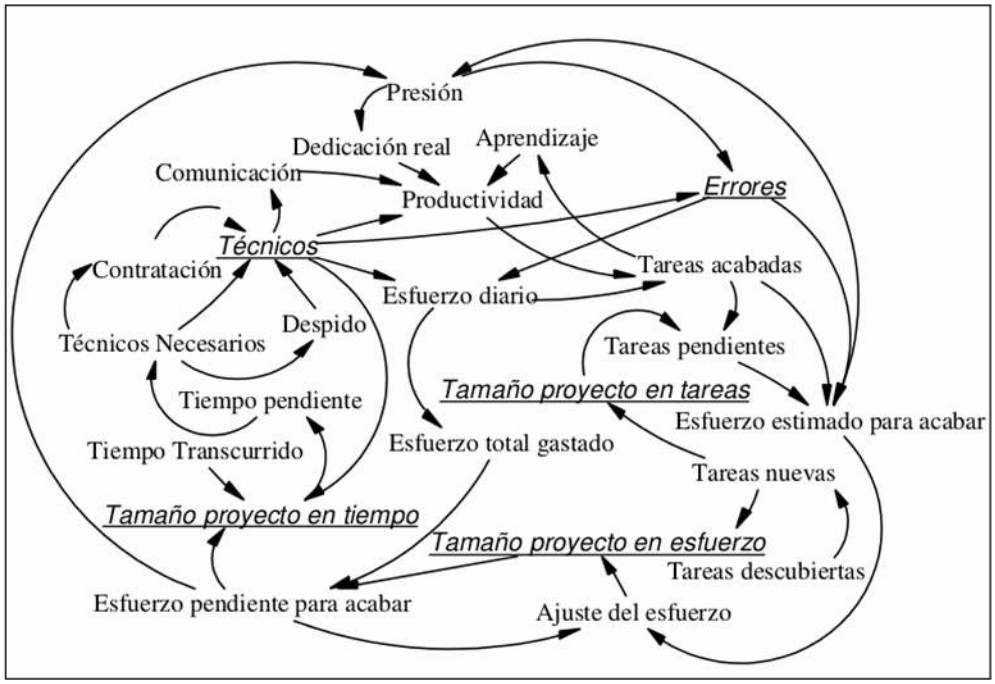


Figura 14 - Modelo Dinámico Reducido [37]

3.7 Conclusiones

En este capítulo se han presentados seis trabajos que se relacionan con esta tesis. En el caso del trabajo presentado en la sección 3.1 el mismo representa el primer esfuerzo por modelar mediante dinámica de sistemas un proceso de desarrollo de software, que en ese entonces era utilizando metodologías en cascada. En el caso del trabajo presentado en 3.2 este fue desarrollado por un equipo integrado por el autor de esta tesis con el objetivo representar las prácticas propuestas por la programación extrema, la experiencia adquirida para la elaboración de esta tesis. En este trabajo se han basado el Subsistema de Producción y Desarrollo de Tareas.

Para el caso del trabajo 3.3, el mismo presenta un modelo que se basa en 7 genes comunes a varias metodologías agiles que según el autor son característicos de estas metodologías. Al centrarse en estos siete genes, el modelo gana en generalidad pero pierde en especificidad con respecto a las prácticas y artefactos de *Scrum*. Además no se enfoca en predecir o volver a crear resultados de proyectos reales.

En caso del trabajo 3.4 se presenta como un modelo que representa el desarrollo de proyecto ágil, pero sin especificar la metodología, el mismo se centra en analizar la relación entre el número de errores que pudieran existir durante el desarrollo y la presión existente respecto al plazo para la finalización de un proyecto. El mismo ha sido utilizado de inspiración para la construcción de los Subsistemas de Presión en el Plazo y de Pruebas de Desarrollo y de Pruebas de integración. Cabe destacar que en este trabajo solo se utilizaron casos hipotéticos ideales para las pruebas realizadas.

En la sección 3.5 se presenta un modelo para una metodología ágil denominada CFMethod, con lo cual no se ajusta a las prácticas y artefactos de *Scrum*. Además este trabajo centra su atención en los costos del proyecto principalmente en cuanto su mantenimiento, lo cual excede el alcance de esta tesis. De todas maneras este trabajo

presenta interacciones entre variables que fueron utilizadas en el Subsitema de Presión en el Plazo.

En trabajo de la sección 3.6 ha servido de base para el desarrollo de esta tesis, de él se han tomado Subsistema de Desarrollo de Tareas, Subsistema de Producción importantes para la realización de esta tesis. Lo que diferencia este trabajo es la ampliación y adaptación del mismo a la metodología ágil *Scrum*.

Cabe destacar que este trabajo se diferencia de los antes mencionados por las siguientes características:

- El modelo está desarrollado específicamente para *Scrum*.
- Tiene en cuenta pruebas de Integración y Desarrollo.
- Incorpora el Subsitema de Planificación como contenedor de todo un proyecto.
- Divide el Subsitema de Recursos Humanos en varios Subsistemas, dando más alternativas de la gestión de los mismos a los gerentes de proyectos.
- Separación de *Corncers*, Cada subsistema puede evolucionar independientemente.
- Fue validado con proyectos reales.
- El modelo cuenta con más de 150 variables.

4 Diseño del Modelo de Estructura del Simulador

A partir de este capítulo comienza el Diseño del Simulador Dinámico de Proyectos de Desarrollo de Software que utilizan Metodología Scrum, aporte de este trabajo. El mismo se construyó siguiendo las fases propuestas por la metodología de Dinámica de sistemas, explicada en la sección 1.4. Estas fases se desarrollaran en este y los siguientes capítulos.

En este capítulo se presentan y se detallan los modelos de estructura propuestos. Estos modelos corresponden a la Fase de Conceptualización. Aquí se explica y se describe el funcionamiento de los diferentes diagramas de influencia de los Subsistemas obtenidos a partir del analizar las fases, artefactos y roles de la metodología Ágil Scrum.

En algunos de los subsistemas modelados, se han identificado diferentes arquetipos sistémicos los que serán también descriptos y detallados.

4.1 Diagrama de Influencias.

El diseño los diagramas de influencia forman parte de la metodología dinámica de sistemas y corresponden a la fase de conceptualización.

Dada la complejidad del sistema y teniendo como base los trabajos de un Modelo Dinámico de Simulación de Proyectos de Software con XP [5] y un Modelo Dinámico Reducido [37] el sistema modelado se dividió en diferentes Subsistemas, de acuerdo con [21] los cuales se enumeran a continuación.

- Planificación (Pregame Phase)
- Producción (Development Phase)
- Desarrollo de Tareas (Development Phase)
- Pruebas de Desarrollo (Development Phase, PostGame Phase)
- Pruebas de Integración (PostGame Phase)
- Presión en el Plazo (Todas las fases)
- Desarrollo de Tareas Extras (Development Phase)
- Promociones de R.H. (Todas las fases)
- Experiencia de R.H. (Todas las fases)
- Cansancio de R.H. (Todas las fases)
- Horas Trabajadas de R.H. (Todas las fases)
- Inasistencias de R.H. (Todas las fases)
- Variables auxiliares y Constantes

Cada modelo participa principalmente en una o más fases de las propuestas por la metodología Scrum vistas en la sección 2.4.4.

En la Figura 15 se presenta una visión global del diagrama influencia de los distintos Subsistemas y sus relaciones.

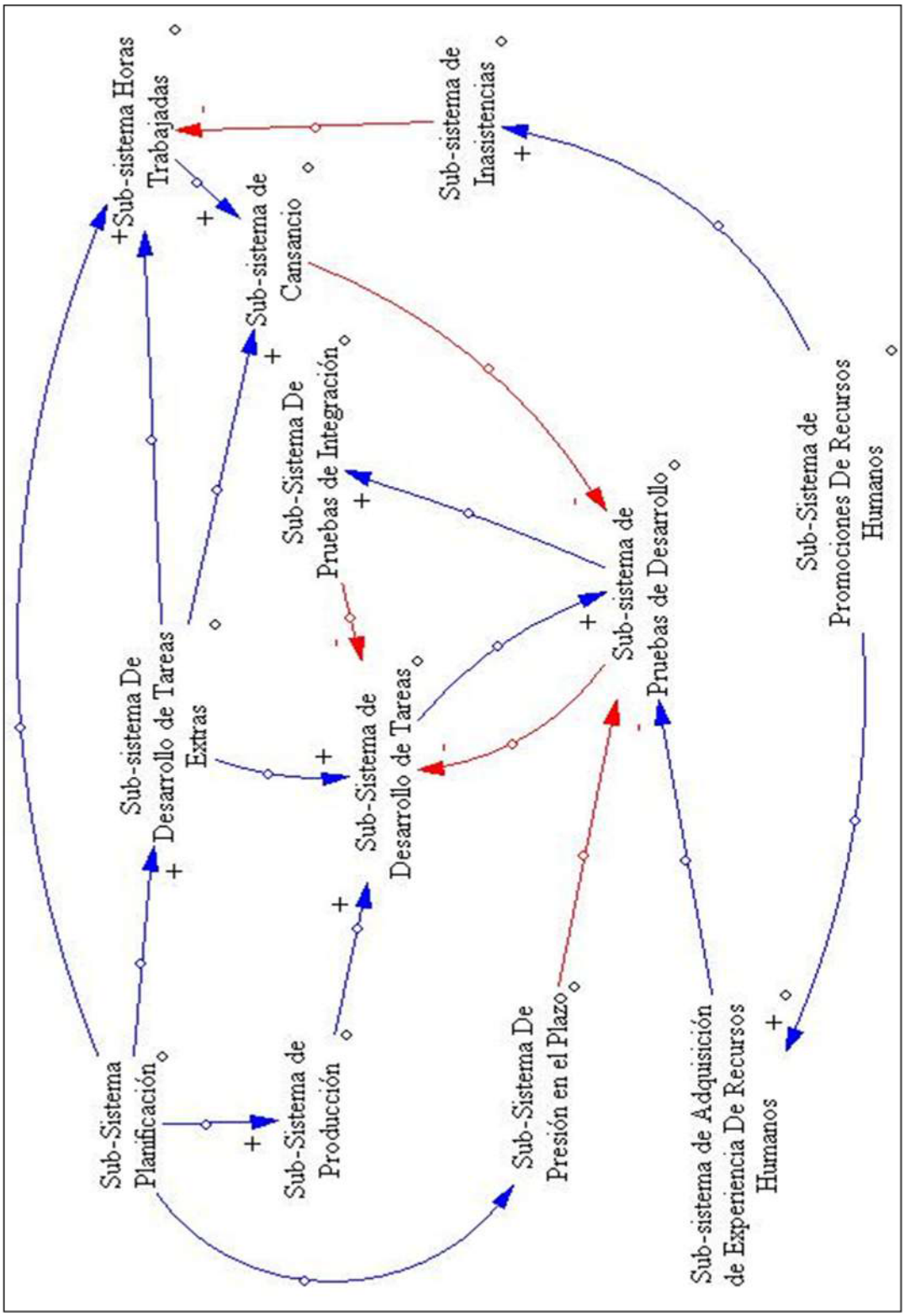


Figura 15 –Diagrama de Influencias entre de Subsistemas

Si bien la división en subsistemas y la asignación tipos de variables intervinientes, son propias a la fase de formulación de la metodología de dinámica de sistemas, dado que la metodología mencionada es iterativa e incremental, este capítulo se presentan los diagramas de influencia de los subsistemas resultantes luego de finalizar el trabajo, habiendo cumplido con todas las fases reiteradas veces hasta que se logró un modelo consolidado.

Para una mejor compresión de los diagramas siguientes se han utilizado prefijos y minúsculas/mayúsculas, para diferenciar entre tipo de variables en los diagramas. La nomenclatura se puede ver en la Tabla 2.

Tabla 2 – Prefijos y Significados de Variables

Prefijo	Significado
lkp	Tipo de Varible Lookup.
aux	Variable Auxiliar
flj	Variable de Flujo
MAYUSCULAS	Variable de Nivel
minúsculas	Parámetros.

Hechas estas aclaraciones se describirán a continuación los subsistemas mencionados.

4.2 Subsistema de Planificación

En el Subsistema presentando en la Figura 16 se realizan cálculos y se establecen valores a los parámetros del modelo por parte del Scrum Master que representan la base para poder iniciar el proceso de simulación. Dentro de los parámetros que especifica el Scrum Master están: el inicio, la duración, la cantidad de puntos de historia a realizar y la velocidad ideal estimada de trabajo diario para cada Sprint.

En el Subsistema se determinan los valores de los parámetros que determina el instante de inicio y duración de cada uno de los Sprints, tal es el caso de las variables lkpSprintsInicio y lkpSprintsDuracionAcumulada.

Otra de las variables es lkpVelocidadPorSprint que toma su valor a partir de la división de los puntos por Sprint y la duración de cada uno de estos. El valor de este parámetro se considera como el más próximo a la velocidad ideal para el desarrollo de los puntos en cada uno de los Sprints.

La variable SPRINTSPLANIFICADOS almacena el número de Sprint en curso, en la medida que el tiempo avanza y coincide con el establecido en la variable lkpSprintInicio toma diferentes valores que se corresponden con el número de Sprint.

En forma conjunta las variables SPRINTSPLANIFICADOS y lkpVelocidadPorSprint determinan el valor de auxVelocidadIdeal que se utiliza para determinar la velocidad de trabajo diario correspondiente a cada Sprint. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

La variable lkpPuntosPorSprint permite al Scrum Master y al Team establecer la cantidad de puntos que deberán completarse en cada uno de los Sprints.

Otra de las variables del modelo es auxIniSprints que muestra el instante del tiempo en el cual se inicia un nuevo Sprint, su valor viene dado por las variables: lkpSprintsInicio que contiene el momento donde da inicio cada Sprint, con la variable Time y con SPRINTSPLANIFICADOS.

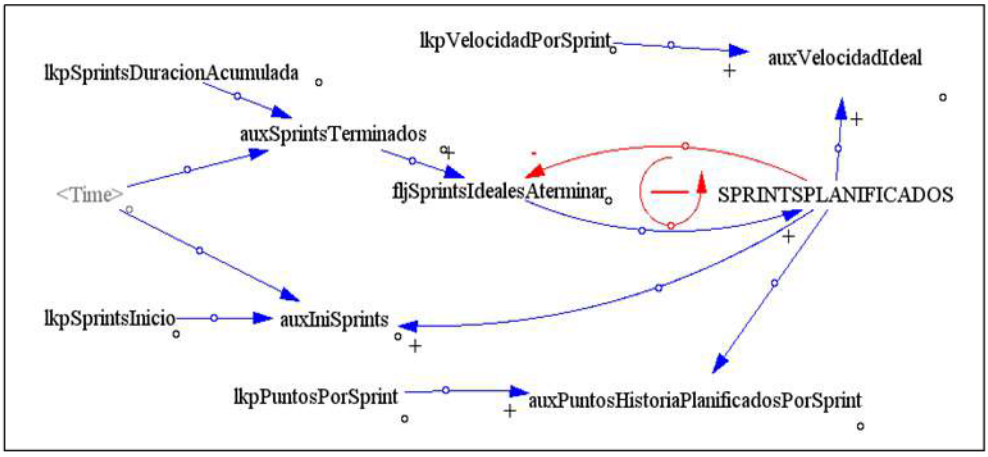


Figura 16 – Diagrama de influencias Subsistema Planificación

4.3 Subsistema de Producción

En la Figura 17 se puede ver el diagrama de influencias correspondiente al Subsistema de Planificación. Este subsistema recoge la dinámica del trabajo realizado al momento de definir la cantidad de Puntos de Historia a incluir en el Sprint y de cómo estos se completarán bajo condiciones ideales.

La variable que da inicio al comportamiento del Subsistema es PUNTOSPORSPRINT que se genera a partir de las variables donde se establecen los puntos de historia determinados y el momento del inicio de cada Sprint.

Con los valores establecidos en PUNTOSPORSPRINT se genera la variable BURDOWNCHART, la cual también depende del valor que asuma auxIniSprint y auxCalculaReqsPorHacer. Su valor se establece a partir de la cantidad de puntos que el Team estima serán los ideales para el Sprint luego de la reunión de Sprint Planning (sección 2.4.2). A estos puntos de historia estimados se le van restando los puntos de historia que se realizarían diariamente y que se establecen en la velocidad inicial estimada de trabajo.

Otra de las variables es PUNTOSPORHACER donde se representan los puntos de historia que se irían completando en la medida que el tiempo avanza. El contenido de la variable se origina a partir de lo estimado en BURDOWNCHARTS y del valor de auxCalculaPuntos. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

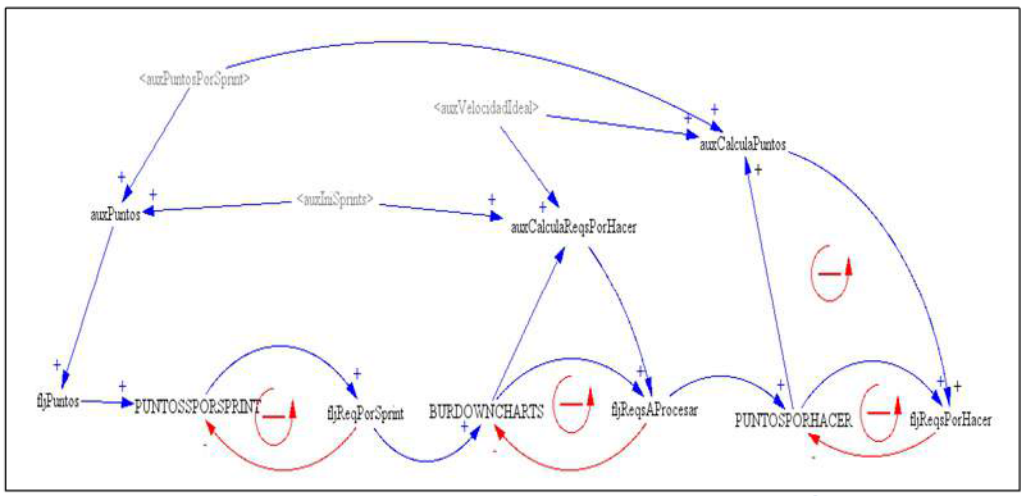


Figura 17 - Diagrama de Influencias del Subsistem Producción

4.4 Subsistema de Desarrollo de Tareas

En la Figura 18 se presenta el subsistema de Desarrollo de Tareas, el cual muestra la dinámica del trabajo realizada al inicio de cada Sprint en lo relacionado al Diseño y Codificación de tareas.

Este Subsistema tiene origen en Subsistema de Producción (sección 4.3), y considerando un número de tareas fijo para cada punto de historia se calcula el total de Tareas Planificadas para cada Sprint representado en la variable TAREASPLANIFICADAS,

La variable que representa las Tareas Planificadas sirve de entrada a la variable TAREASPENDIENTESCODIF, que son aquellas tareas que están pendientes de codificar, en la medida que las tareas pendientes se codifican el avance del trabajo se reflejará en una tercera variable llamada TAREASCODIFICADAS.

La variable TAREASCODIFICADAS cuyo contenido además de ser establecido por la cantidad de tareas iniciales a codificar, se incrementa mediante la variable auxTareasARecodificar que suma las tareas con errores de codificación o con errores de integración provenientes de los Subsistemas presentados en las sección 4.5 y la sección 4.6. Otro factor de incremento de TAREASCODIFICADAS son las diferentes tareas planificadas a lo largo del Team, el único caso donde esta variable es afectada de manera negativa es cuando se produce la ausencia de algún integrante del Team, en cuyo caso el desarrollo normal de las tareas se reduce.

La variable TAREASCONERRORES permite visualizar la totalidad de tareas con errores de diferentes tipos que se van dando en cada uno de los Sprints.

Una variable relevante en este Subsistema es tasaAjusteVelocidad, la cual permite aumentar o disminuir de manera porcentual e igualitaria la velocidad de desarrollo de las tareas para los diferentes Sprints. Dado que pueden aparecer tareas extras o tareas a recodificar la velocidad inicial estimada no sería suficiente para poder completar el conjunto de tareas pendientes, entonces en la medida que el factor de ajuste la velocidad de trabajo aumente o disminuya la cantidad de tareas codificadas aumentará o disminuirá en función de esta. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

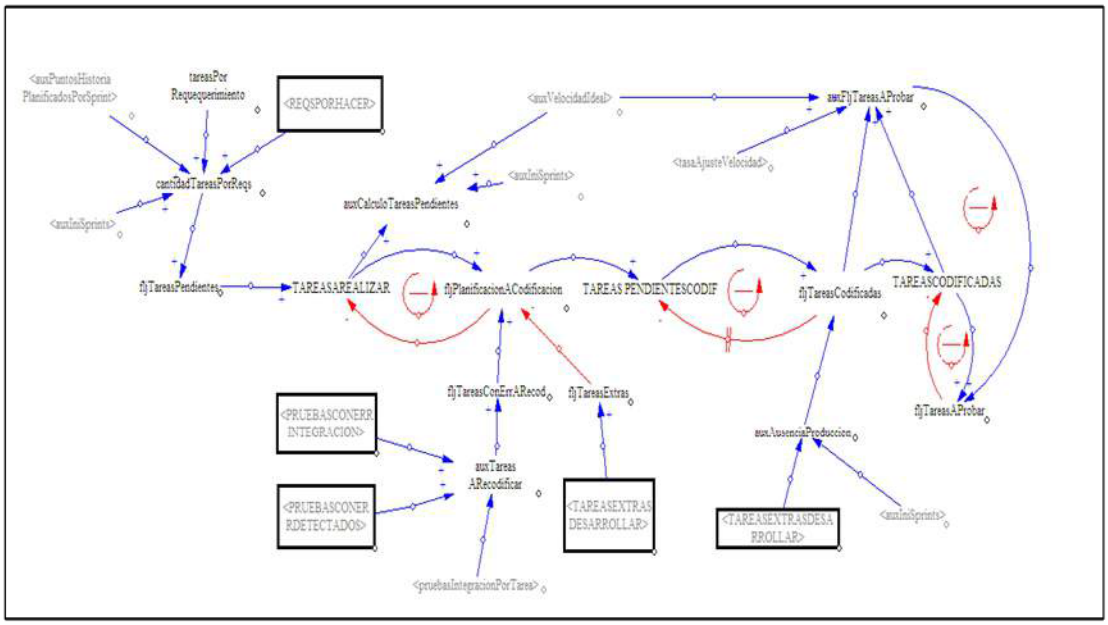


Figura 18 - Diagrama de influencias del Subsistema Desarrollo de Tareas

4.5 Subsistema de Pruebas de Desarrollo

El Subsistema presentado aquí, permite modelar el comportamiento donde se llevan a cabo diversas pruebas que permiten detectar el número de tareas codificadas que presenten errores de lógica o errores de codificación.

El Subsistema comienza a partir de un conjunto de pruebas a realizar que resulta del producto entre un número de pruebas por tarea y la cantidad de tareas codificadas que se generaron en el subsistema explicado en la sección 4.4. La cantidad de pruebas a realizar y que tendrá efecto directo sobre el comportamiento del Subsistema las determina el Scrum Master.

Una vez diseñadas las pruebas se transforman en la entrada de otra variable que es PRUEBASAREALIZAR y representa el número total de pruebas que se llevarán a cabo para verificar la existencia o no de errores de desarrollo.

Con el total de pruebas a realizar determinado y en función de una tasa de errores por Presión en el Plazo y una tasa de errores producidos por el Cansancio se determinan dos nuevas variables, la primera PRUEBASCONERRDETECTADOS que contiene el número de tareas con errores lógicos producidos, y otra con las tareas que no poseen errores de lógica denominada PRUEBASSINERROR.

Finalmente con los valores de la variable PRUEBASINERROR se calcula el número total de pruebas de integración a realizar. Esta cantidad de pruebas a realizar resultan del producto de multiplicar la cantidad de pruebas sin error por un número de pruebas de integración por cada tarea. El diagrama causal del Subsistema explicado en este apartado se representa en la Figura 19. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

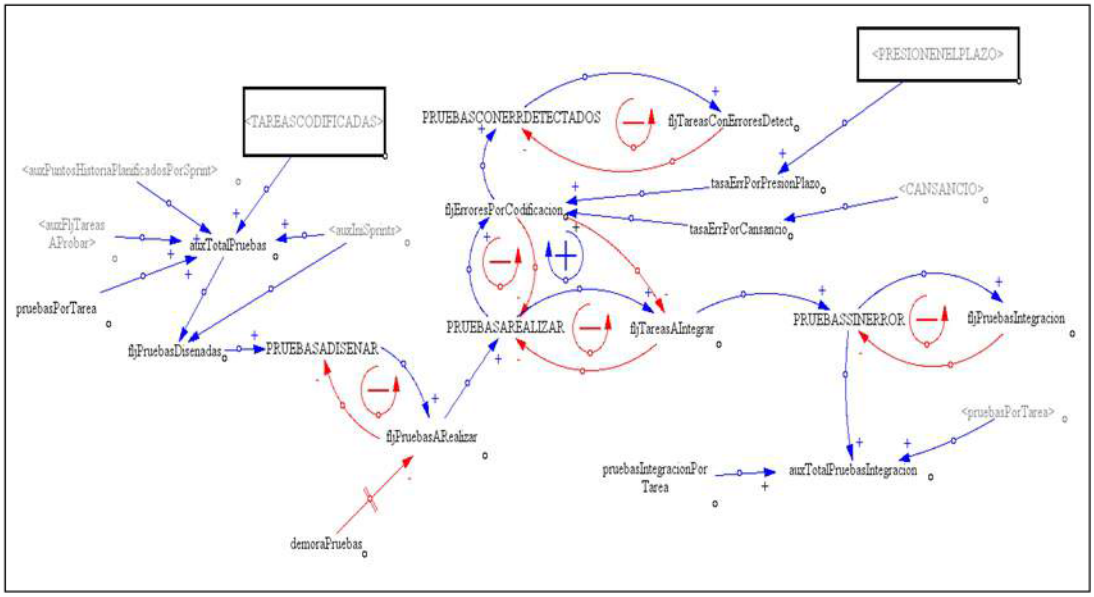


Figura 19 - Diagrama de influencias del Subsistema Pruebas de Desarrollo

4.6 Subsistema de Pruebas de Integración

El Subsistema que se presenta en este apartado, expresa la lógica previa a la integración final de las diferentes tareas codificadas. El Subsistema tiene su origen en el Subsistema de Control de Errores de Desarrollo más precisamente en aquellas tareas que no han tenido errores lógicos y ni de codificación, así, estas tareas programadas son sometidas a ciertas pruebas de integración.

El Subsistema se inicia a partir de la variable auxTotalPruebasIntegracion generada en Subsistema de Pruebas de Desarrollo en 4.5, y es el resultado de la multiplicación de la cantidad de tareas por un número de pruebas fijo e igual para todas las tareas codificadas.

A partir de la variable auxTotalPruebasIntegracion se genera una nueva variable llamada PRUEBASINTEGRACIONDISENO que almacenará la cantidad de pruebas de integración que deberán realizarse.

La siguiente variable del Subsistema es PRUEBASINTEGRACIONREALIZADAS, y representa la totalidad de las pruebas que se realizaron. Con las pruebas de integración terminadas y mediante una tasa de error que selecciona el Scrum Master, se generan dos nuevas variables PRUEBASINTEGRADAS que almacenará las tareas que pasan las pruebas de integración y PRUEBASCONERRINTEGRACION que representa aquellas tareas que poseen errores de integración, y que por lo tanto deberán reprogramarse en el Subsistema presentado en la sección 4.4.

El diagrama causal del Subsistema explicado en este apartado se presenta en la

Figura 20. Desde el punto de vista sistémico este subsistema cuenta con cinco bucles de retroalimentación negativa.

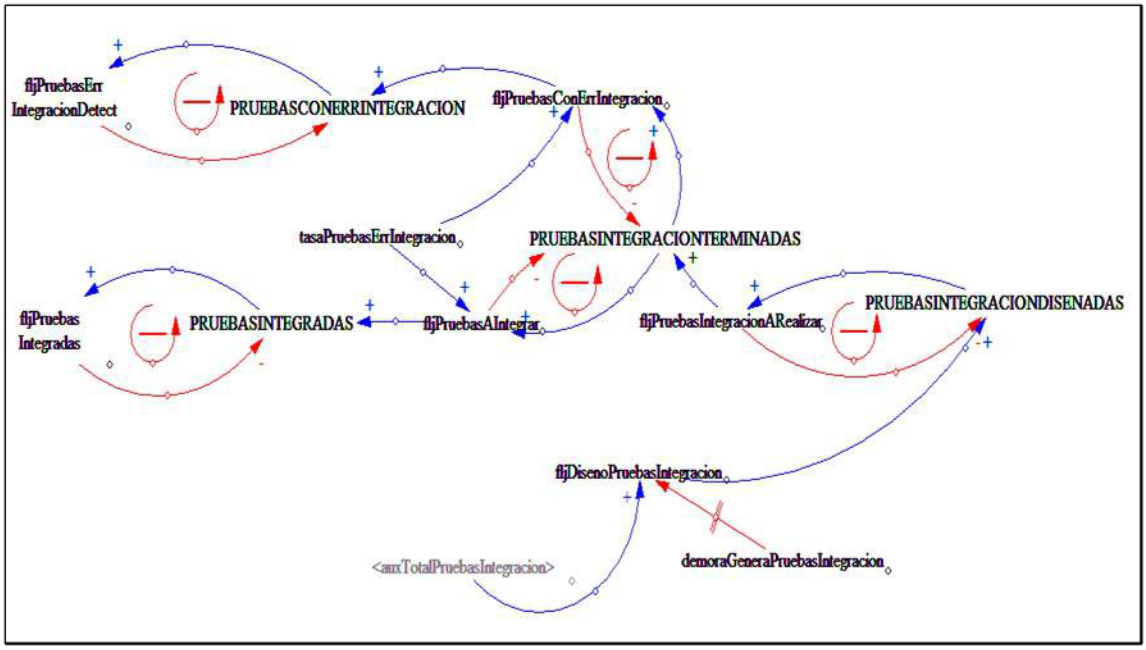


Figura 20 - Diagrama Causal Subsistema de Pruebas de Integración

4.7 Subsistema de Presión en el Plazo

El Subsistema de Presión en el Plazo que se aprecia en la Figura 21 modela la presión que sufre el Team en la medida que transcurre el tiempo durante el desarrollo del proyecto y en la medida que la finalización del mismo se acerca.

En la medida que el tiempo transcurre el plazo acordado para finalizar es cada vez menor y esto hace que aumente la presión. Este aumento de presión está dado por la variable lkpPresionPlazo que representa un factor o tasa de presión que sufriría el Team en la medida que se acerca la finalización del proyecto, y cuyo valor es establecido por el Scrum Master en función de su experiencia.

La variable PRESIONPLAZO representa la presión total que el Team sufre a lo largo del proyecto. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

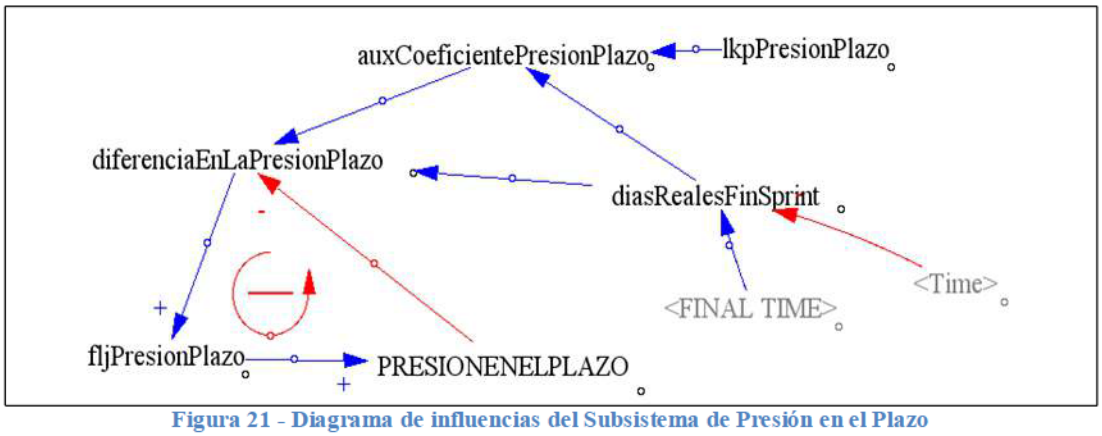


Figura 21 - Diagrama de influencias del Subsistema de Presión en el Plazo

4.8 Subsistema de Desarrollo de Tareas Extras

Este Subsistema modela las tareas extras que pudieran surgir por errores o fallas en el análisis de requerimientos realizados al inicio del Sprint y que son detectadas una vez iniciado el Sprint, más aquellas tareas extras planificadas por el Scrum Master y el Team.

Las tareas que deben ser desarrolladas se acumulan en la variable TAREASEXTRASDESARROLLAR y son el resultado de sumar las tareas planificadas y las no planificadas.

En relación a estas últimas la probabilidad de que pudieran surgir la administra el Scrum Master mediante la modificación del valor de la variable probabilidadTareaExtraNoPlanificada

La variable lkpTareasExtrasSprint representa un conjunto de tareas extras que el Team debe desarrollar durante el proyecto y que se conocen de manera previa al inicio del Sprint.

Finalmente la variable TAREASEXTRASREALIZADAS presenta la acumulación de tareas extras a lo largo del proyecto.

El Subsistema explicado en este apartado se representa en la Figura 22. Desde el punto de vista sistémico este subsistema cuenta con un bucle de retroalimentación negativa.

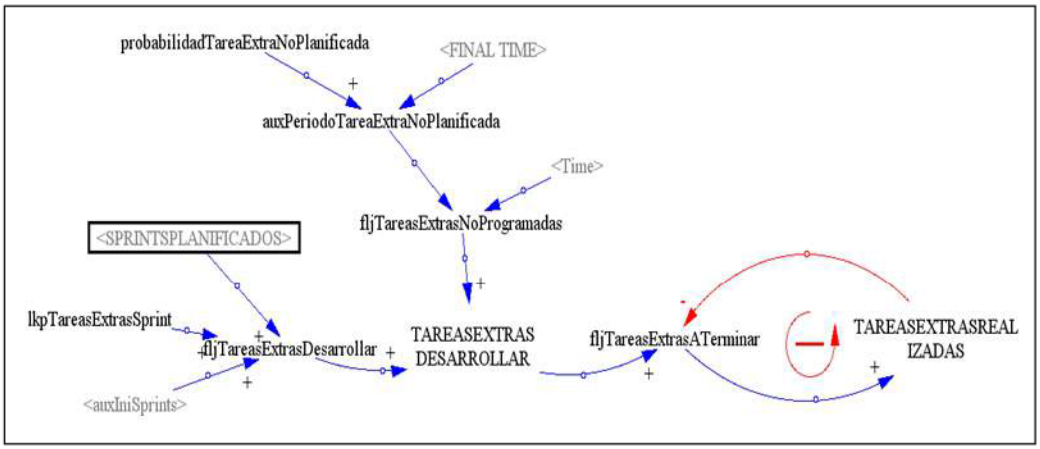


Figura 22 - Diagrama de influencias del Subsistema de Desarrollo de Tareas Extras

4.9 Subsistema de Búsqueda y Reclutamiento de Recursos Humanos

En la Figura 23 se presenta el Subsistema de Recursos Humanos correspondiente a la Búsqueda, Selección y Reclutamiento de personas que integrarán el Team.

Este Subsistema representa el proceso de búsqueda y contratación de personas a partir de la diferencia existente entre el número ideal de integrantes del Team definidos al inicio del proyecto y de la cantidad de integrantes reales del Team hasta lograr que el número ideal de integrantes sea igual al número de integrantes reales del Team.

En caso de que la diferencia en la cantidad de integrantes sea diferente a cero, y se considere necesario, se procede a la búsqueda de un número de reemplazantes que complete la cantidad ideal de integrantes.

La búsqueda de un nuevo integrante puede producirse de manera inmediata o bien luego de cierto tiempo. El número de candidatos preseleccionados se establece de manera aleatoria entre dos valores. Los valores que adopten ambas variables dependerán de la experiencia del Scrum Master.

Una vez realizada la preselección de candidatos, representados en la variable RHPRESELECCIONADOS y en función de la necesidad del Scrum Master de incorporar integrantes Seniors o Juniors se establece el valor de la variable porcentajeSeniorsBuscados.

Los candidatos Seniors o Juniors que son preseleccionados se representan en las variables RHCANDIDATOSSENIORS y RHCANDIDATOSJUNIORS. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

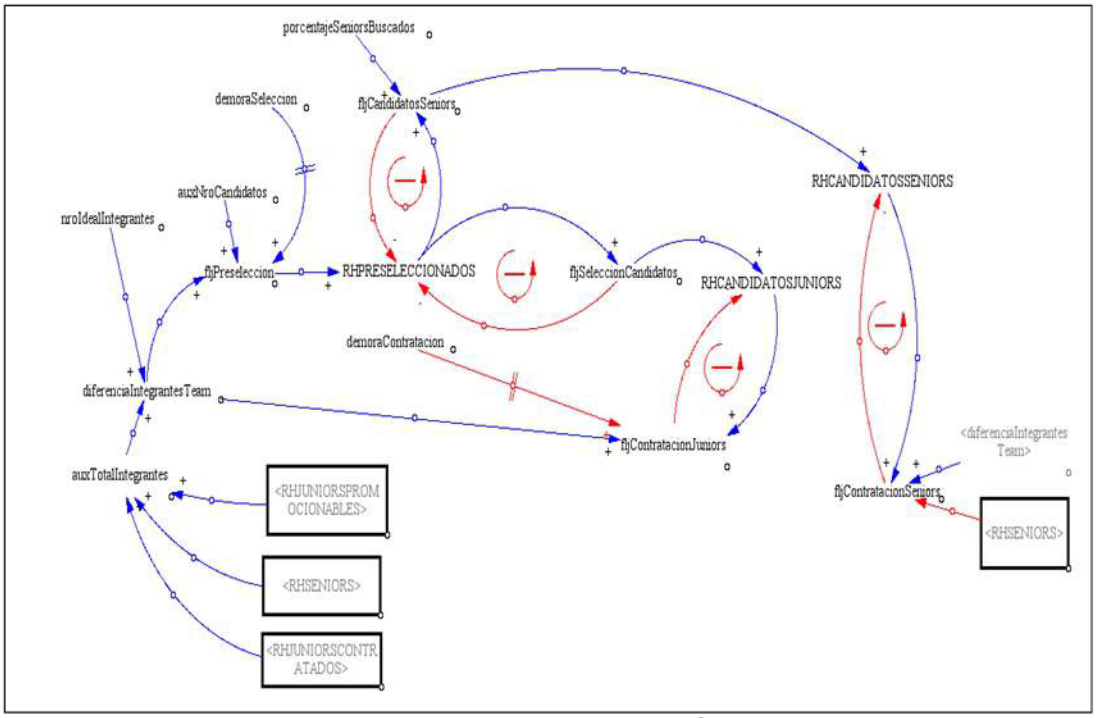


Figura 23 - Diagrama de influencias del Subsistema Búsqueda y Reclutamiento

4.10 Subsistema de Promoción de Recursos Humanos

En la Figura 24 se presenta el Subsistema de Promoción de Recursos Humanos de los integrantes del Team. Se consideran aquí los niveles de experiencia Juniors, Juniors Promocionales y Seniors a partir del trabajo presentado en [38].

El Subsistema está vinculado directamente con el Subsistema de Búsqueda y Reclutamiento, ya que este último provee de las entradas necesarias a las variables RHSENIORS y RHCANDIDATOSJUNIORS. Cuando se produce la contratación de un nuevo integrante y este no posee la experiencia suficiente en la metodología se lo considera Junior Contratado, y en la medida que se considere oportuno es promovido a los siguiente niveles de experiencia.

La variable RHJUNIORSPROMOCIONABLES representa a los integrantes que están en condiciones de convertirse en Seniors dentro del Team luego de cierto tiempo.

El Subsistema permite al Scrum Master administrar los tiempos para que las promociones en los diferentes niveles de experiencia se sucedan, pudiendo además establecer cuantos Juniors contratados pasaran a nivel de JuniorsPromocionales, y que cuántos de estos últimos serán promovidos a Seniors. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

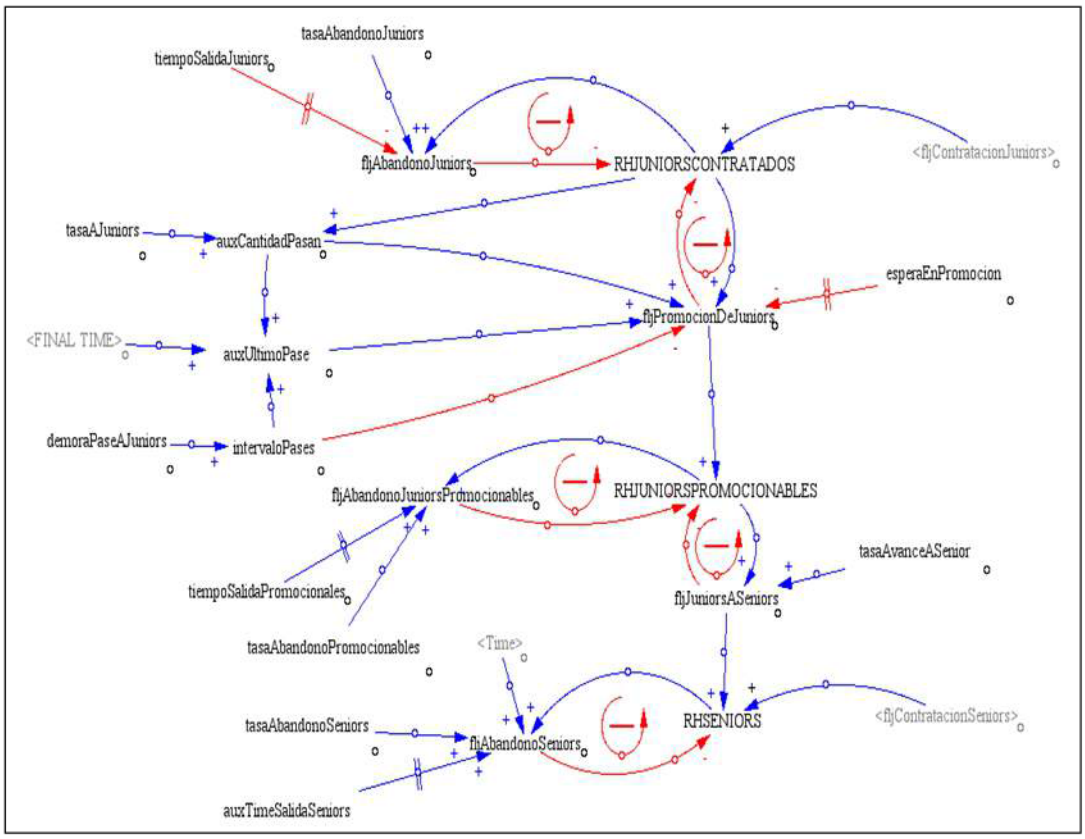


Figura 24 - Diagrama de influencias del Subsistema Promoción

El resto de las variables y parámetros permiten al Scrum Master establecer los instantes de tiempo en que se producirán abandonos en cada uno de los niveles de experiencia contemplados.

4.11 Subsistemas de Adquisición de Experiencia

El Subsistemas presentado en este apartado modela la adquisición de experiencia del Team a lo largo del proyecto.

La acumulación total de la experiencia se presenta en la variable EXPERIENCIATEAM, y es la resultante de ponderarla experiencia de los integrantes y de una tasa de aprendizaje que es establecida por el Scrum Master. En la medida que las tasas de aprendizaje sean más altas, más rápido el equipo adquirirá experiencia.

Este Subsistema está vinculado al Subsistema de Presión en el Plazo y al de Recursos Humanos, ejerciendo influencia sobre el primero y siendo afectado por la cantidad de integrantes que se modele en el segundo.

El Subsistema explicado en este apartado se representa en la Figura 25. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

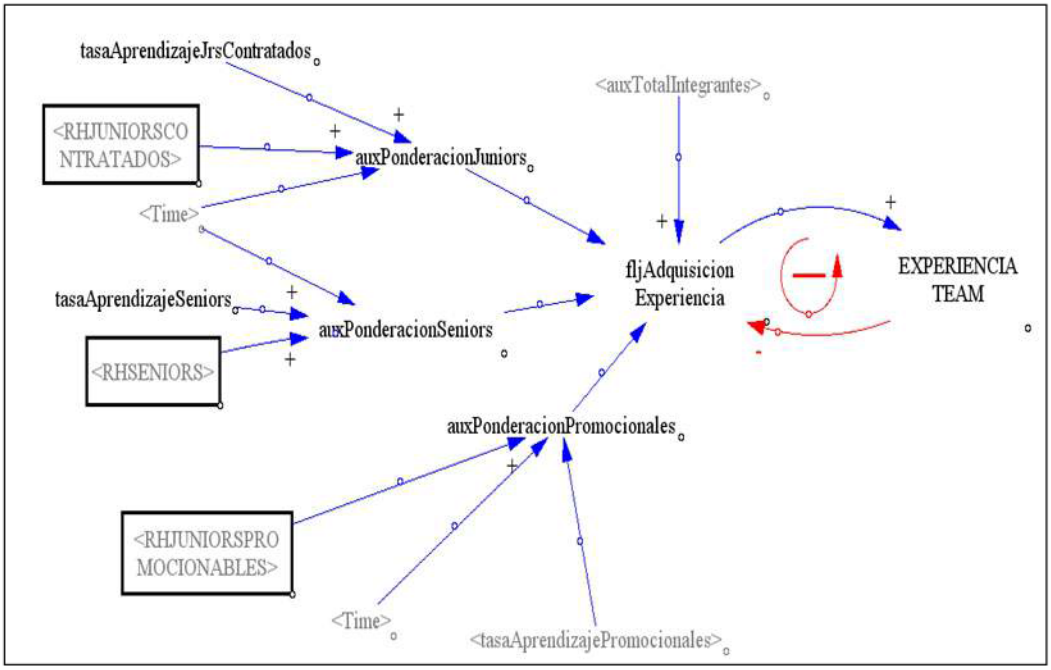


Figura 25 - Diagrama de influencia del Subsistema de Adquisición de Experiencia

4.12 Subsistema de Cansancio

El diagrama causal de este apartado modela el cansancio que sufre el Team en la medida que avanza el tiempo de desarrollo del proyecto. El comportamiento viene dado por dos variables: lkpCoeficienteCansancioDiario y lkpCoeficienteCansancio.

En la primera variable mencionada el Scrum Master determina la tasa de cansancio que estima tendrán los integrantes del Team en función de la cantidad de horas diarias trabajadas. El valor de la misma surge de la suma de horas normales trabajadas más las horas extras.

La segunda variable también establece una tasa de cansancio, pero en este caso viene dada por el porcentaje de horas de trabajo que el Team lleva en el proyecto. Esta tasa también es establecida por el Scrum Master previo al inicio del proyecto.

Por su parte, la variable CANSANCIO almacena el cansancio total del Team a lo largo del proyecto. El diagrama causal correspondiente al Subsistema de cansancio se presenta en la Figura 26. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

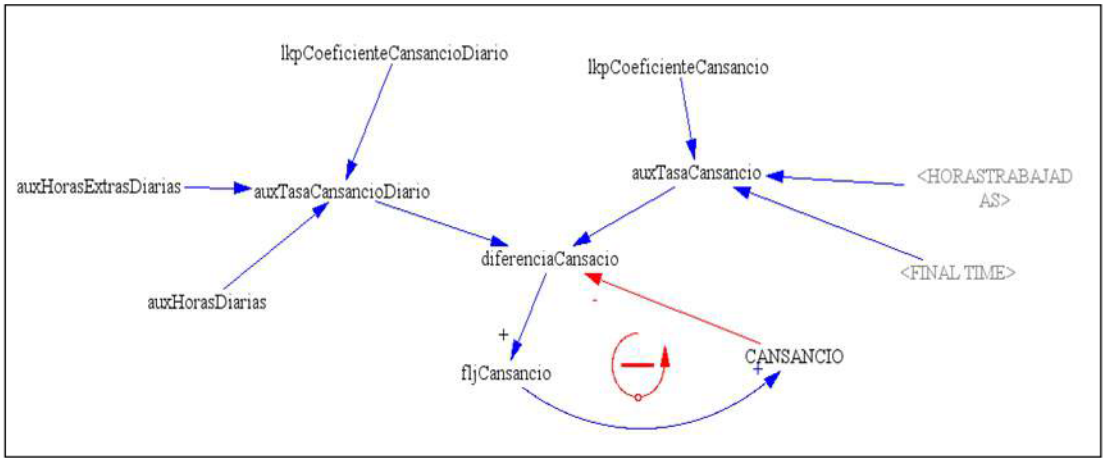


Figura 26 - Diagrama de influencias del Subsistema de Cansancio

4.13 Subsistema Horas Trabajadas

El diagrama causal presentado en la Figura 27 modela y representa el total de horas trabajadas por el Team a lo largo del proyecto.

Las horas totales trabajadas surgen de la suma de horas normales y de horas extras diarias trabajadas por el Team durante el proyecto.

La variable principal en este modelo es HORASTRABAJADAS ya que almacena la totalidad de horas trabajadas a lo largo del proyecto. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

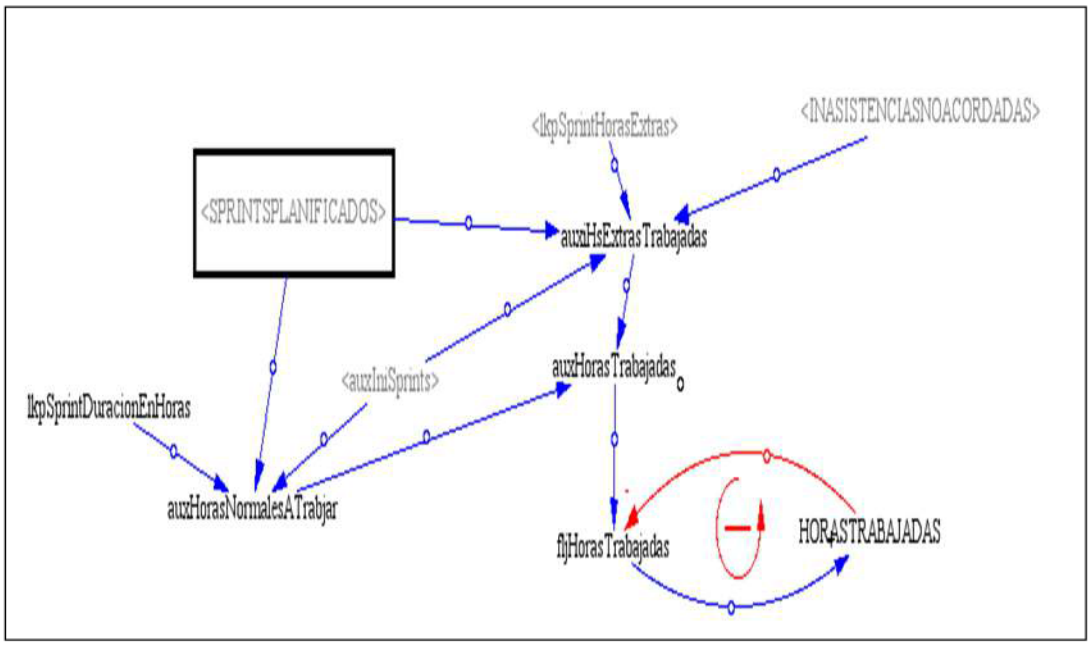


Figura 27 - Diagrama Causal del Subsistema de Horas Trabajadas

4.14 Subsistema de Inasistencias

Este Subsistema modela la generación de Inasistencias no acordadas por parte de los integrantes del Team con el Scrum Master.

El comportamiento del Subsistema se inicia a partir de la probabilidad de que una inasistencia ocurra. Dicha probabilidad puede ser establecida por el Scrum Master a través de la variable probabilidadInasistenciaNoAcordada, y en la medida que dicho valor se modifique el periodo en cual una inasistencia se genera también se irá modificando, provocando esto una mayor o menor cantidad de inasistencias dependiendo del valor asignado a la variable.

En la Figura 28 se presenta el diagrama causal del Subsistema descripto en este apartado. Desde el punto de vista sistémico estas variables forman un bucle de retroalimentación negativa y por lo tanto se considera estable.

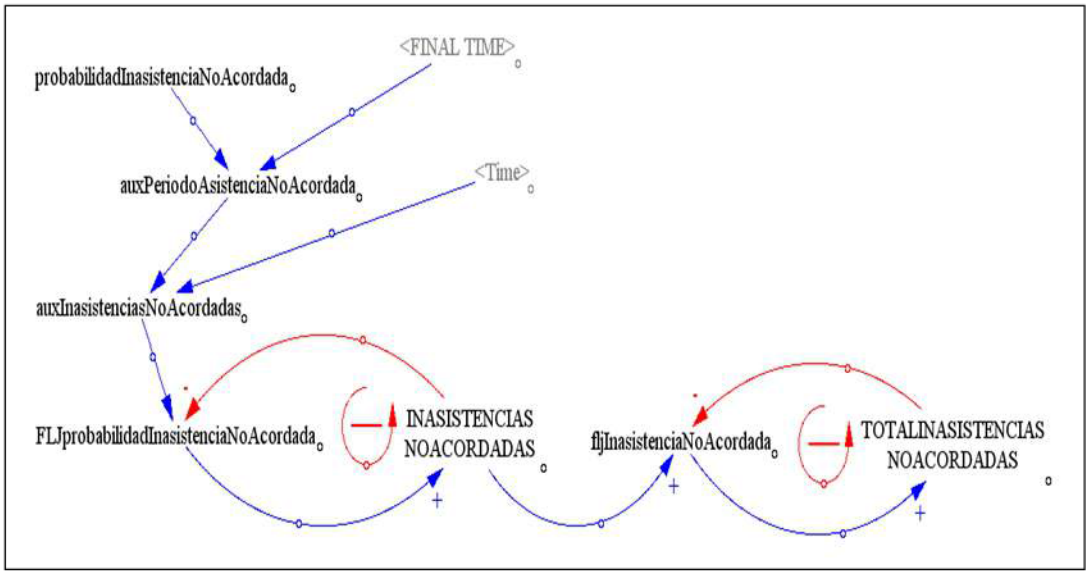


Figura 28 - Diagrama Causal Subsistema Inasistencias

4.15 Arquetipos Sistémicos Identificados

Como se mencionó en la sección 1.3 los sistemas presentan una serie de comportamientos comunes o bien, que aparecen con mayor frecuencia en los mismos.

En este apartado se presentan algunos casos de los diferentes arquetipos sistémicos que se identificaron en el presente trabajo. A continuación se presentan las Figuras de dichos arquetipos y la clasificación de cada uno de ellos.

4.15.1 Límites de Crecimiento Sigmoidal

El arquetipo de Limite de crecimiento Sigmoidal (sección 1.3.1) presenta un comportamiento donde un proceso se alimenta de sí mismo para tener un crecimiento sostenido durante un período de tiempo, pero luego el crecimiento se vuelve más lento y pudiendo llegar a detenerse.

El arquetipo presentado aquí corresponde a los Subsistemas de Búsqueda y Reclutamiento (sección 4.9) y de Promoción (sección 4.10).

En el diagrama solamente contiene los componentes que conforman el arquetipo para una mejor visualización del mismo, los demás componentes fueron ocultados de manera intencional.

En la Figura 29 se presenta el arquetipo descripto.

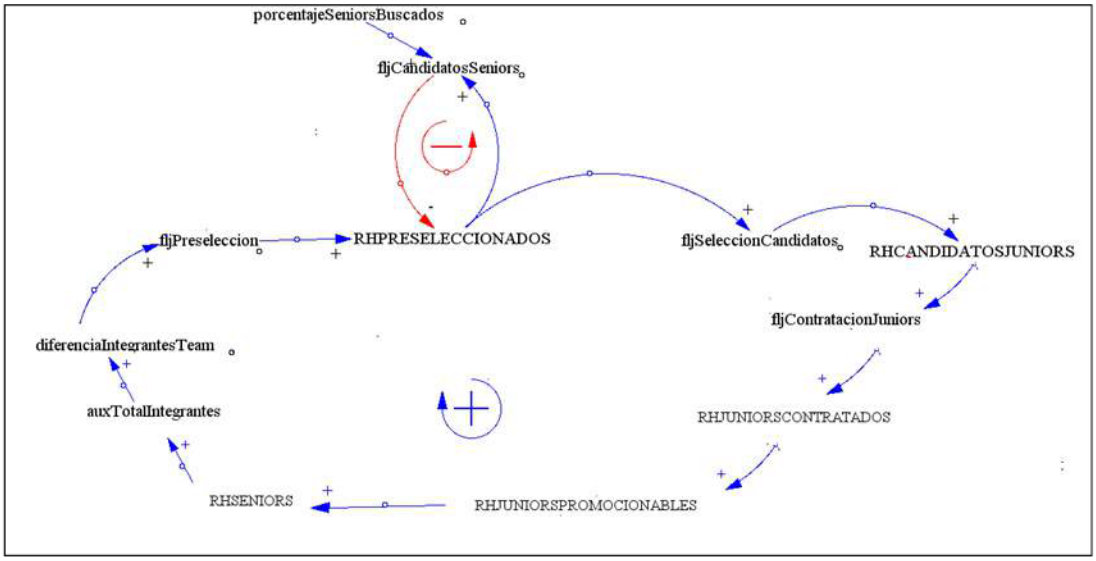


Figura 29 – Arquetipo Limite de Crecimiento Simoidal

Si bien en ciertas situaciones este arquetipo puede resultar contraproducente en un sistema en que se desea que el valor de una variable crezca indefinidamente, en este caso el crecimiento en la cantidad de miembros del team Seniors, estará controlado por el parámetro porcentajeSeniorsBuscados (condición limitante), por lo que el Scrum máster en caso de necesitar poner un porcentaje máximo de Seniors en el Team puede hacerlo. Po el contario si necesita incrementar la cantidad de Seniors en el proyecto solo deberá incrementar el valor de dicha variable para que se comience con el proceso de promoción de Juniors hasta completar el porcentaje deseado.

En este caso el bucle superior limitará el crecimiento indiscriminado de Seniors que se daría si el bucle positivo inferior actuara solo.

4.15.2 Desplazamiento de La Carga

El arquetipo de desplazamiento de carga (sección 1.3.2), que se puede ver en la Figura 30 corresponde al Subsistema de Pruebas de Desarrollo (sección 4.5), en donde se observan las características de este arquetipo, con la participación de las variables fljErroresPorCodificacion, PRUEBASAREALIZAR y fljTareasAintegrar.

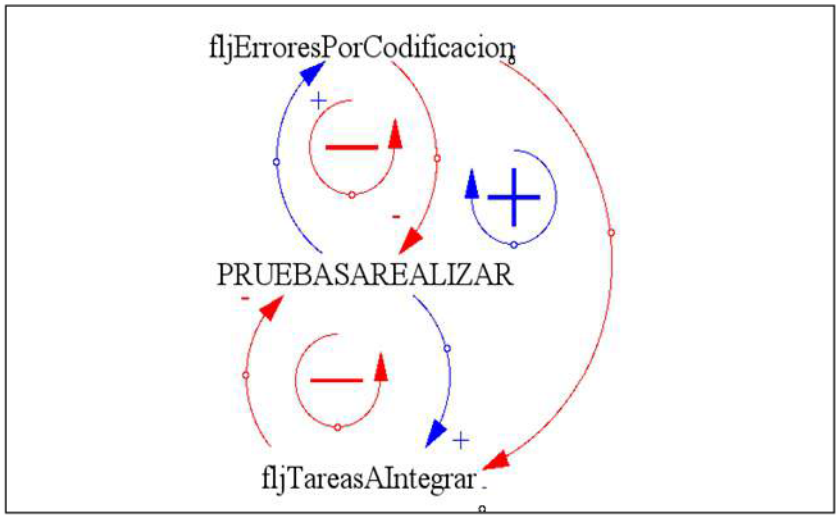


Figura 30 - Arquetipo Desplazamiento de la Carga

El Team en todo momento intentará reducir las PRUEBASAREALIZAR (síntoma) para ello puede reducir las mismas, teniendo Errores por Codificación, es decir

PRUEBASCONERRDETECTADOS (solución Sintomática, bucle negativo superior) o realizándolas correctamente dejando las tareas listas para ser integradas, es decir PRUEBASSINERROR (Solución Fundamental bucle negativo inferior). El problema surge cuando por errores de codificación surgen nuevas pruebas a realizar, disminuyendo la eficacia de las de la solución fundamental de realizar las pruebas sin errores, para disminuir la cantidad de pruebas quedan por realizarse (bucle positivo). Es por ello que se debería Actuar en disminuir los errores de Codificación, que pueden ser inducidos por la presión en el plazo o cansancio.

4.15.3 Compensación Entre Proceso y Demora

El arquetipo de Compensación Entre Proceso y Demora presentado en la sección 1.3.3 corresponde al Subsistema de Desarrollo de Tareas, descripto en la sección 4.4.

Aquí entre las tareas pendientes de codificación y su codificación existe cierta demora, la cual se produce al momento de la codificación de las tareas y al producirse la correspondiente influencia sobre la variable TAREASPENDIENTESCODIF.

El arquetipo y las variables involucradas puede observarse en la Figura 31. En este caso por problemas de comunicación en el team puede llegar a darse un retraso en la diminución de la tareas que se cree faltan codificar, lo que podría derivar en un aumento de la presión por plazo equivocada.

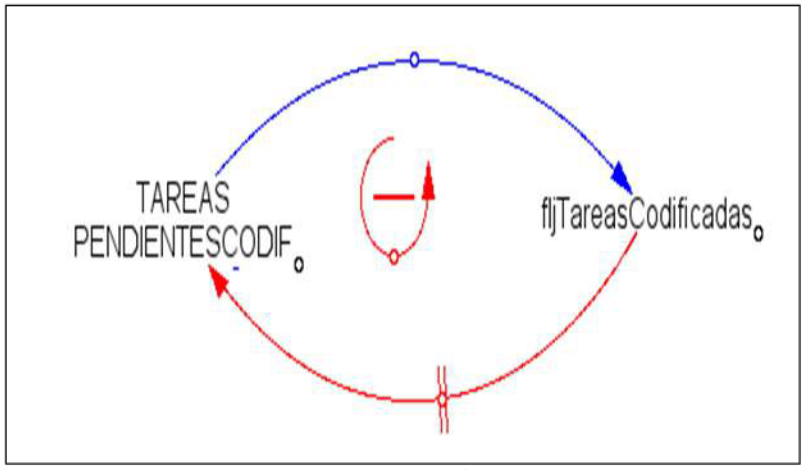


Figura 31 - Arquetipo de Compensación Proceso y Demora

5 Diseño del Modelo de Comportamiento del Simulador

En el presente apartado se presentan los diferentes diagramas de Forrester que se generaron a partir de los diagramas causales modelados en capítulo **¡Error! No se encuentra el origen de la referencia.** y un glosario de variables que tiene como objetivo poder comprender el rol de estas en cada Subsistema y cómo se vinculan e interactúan entre sí.

5.1 Diagramas de Forrester

Estos diagramas en relación a lo propuesto por Forrester [19] corresponden a la fase de Formulación de la metodología dinámica de sistemas (sección 1.4.1). Un concepto nuevo que aparece en este apartado es el de Subsistema Conservativo [21].

Este concepto establece que en un modelo hay que asignarle dimensiones a las variables que lo componen, de manera que todas ellas se midan en determinadas unidades de medida. Si determinados niveles están en una cierta unidad de medida, los flujos que lo complementan deben estar en las mismas unidades a fin de mantener el concepto de válvulas y niveles, de tal manera que lo que entra al nivel sea lo mismo que sale de este.

De esta manera los niveles se asocian entre sí en cascada o en paralelo formando estructuras por las cuales solo circula el mismo tipo de unidades bajo el control de flujos que se miden en esas mismas unidades por unidad de tiempo.

A continuación se presentan los diagramas correspondientes a los Subsistemas identificados y modelados en el capítulo 4 del presente trabajo. Junto a cada uno de los diagramas se presenta un glosario de las variables más relevantes de los mismos, y una explicación sobre la función de cada una.

Vale mencionar que las variables *Shadow* de otros Subsistemas que aparecen en las diagramas de Forrester se explicarán el Subsistema original al que corresponden.

5.2 Subsistema de Planificación

En la Figura 32 se presenta el diagrama de Forrester correspondiente al Subsistema de Planificación, en dicha Figura se pueden apreciar las variables detalladas a continuación.

lkpVelocidadPorSprint: esta variable tipo LookUp, tiene como entrada el número de Sprint y como salida la velocidad ideal estimada para cada sprint que deberá aplicarse para el desarrollo de las tareas planificadas en el sprint. Unidad: puntos.

auxVelocidadIdeal: variable auxiliar donde se aplica la función LOOKUP FORWARD donde a partir del número de Sprint proveniente de SPRINTSPLANIFICADOS, se establece la velocidad especificada en lkpVelocidadPorSprint para un Sprint Determinado. Unidad: puntos/dia.

lkpSprintsDuraciónAcumulada: variable LookUp donde la entrada es el instante del tiempo en el cual finaliza un Sprint y la salida es el número de Sprint al que le correspondería dicho instante de finalización. Unidad: puntos.

auxSprintsTerminados: variable auxiliar donde mediante la utilización de la función LOOKUP FORWARD se establece el número de Sprint vigente a partir del valor de la variable Time. Unidad: sprint.

SPRINTSPLANIFICADOS: variable de nivel utilizada para representar el Sprint vigente en un instante de tiempo dado a largo del proyecto. El valor que toma surge de la variable auxSprintsTerminados. Unidad: sprint

lkpSprintsInicio: es una variable LookUp utilizada para determinar en qué momento comienza cada uno de los Sprints planificados. A partir del número de Sprint, obtiene el valor asociado que representa el instante del tiempo donde se inicia el Sprint. Unidad:hora

auxIniSprints: variable auxiliar que a partir de la variable tipo *lookup*, lkpSprintsInicio y la variable del sistema *Time* generar una serie de pulsos que indican el momento del inicio del Sprint. Unidad: sprint

lkpPuntosPorSprint: esta variable es utilizada para asignar a cada Sprint los puntos de historia que deben completarse en los mismos. El valor de entrada es el número de Sprint, y asociado a esta entrada el número de puntos de historia. Unidad: puntos.

auxPuntosPorSprint: variable que a partir de los valores contenidos en la variable lkpPuntosPorSprint y del número de Sprint presentan en forma de pulsos los puntos a completarse en cada Sprint. Unidad:puntos.

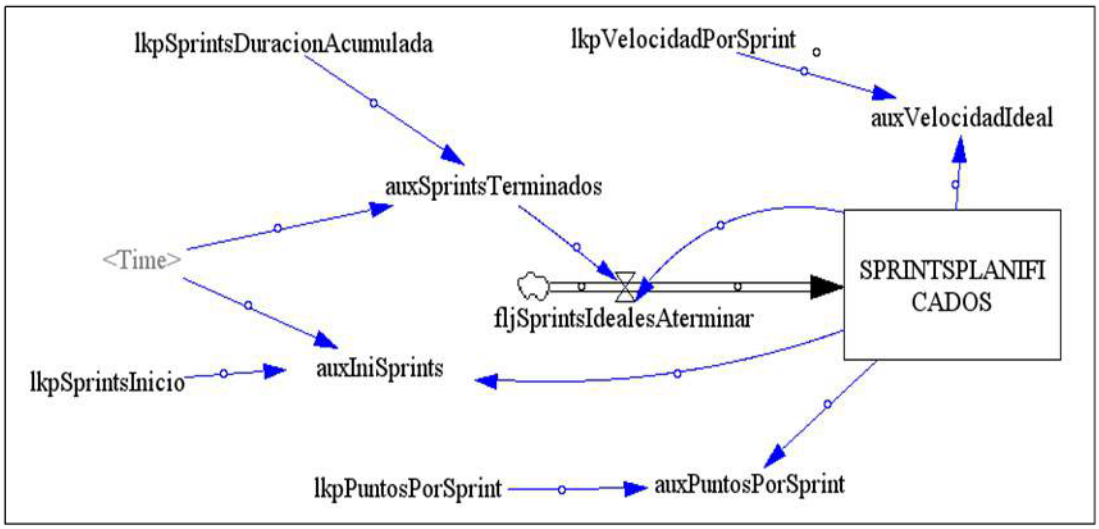


Figura 32 - Diagrama de Forrester Subsistema de Planificación

5.3 Subsistema de Producción

En la Figura 33 se presenta el diagrama de Forrester correspondiente a este Subsistema, y en ella se aprecian las variables detalladas a continuación.

auxPuntos: variable auxiliar que permite determinar la cantidad de puntos que se desarrollarán en cada uno de los Sprints. El contenido de esta variable se determina a partir de los valores establecidos en auxPuntosPorSprint y auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor de los puntos establecidos para dicho sprint. Unidad: puntos.

auxCalculaPuntosPorHacer: esta variable auxiliar permite determinar la velocidad con la cual los puntos asignados a cada Sprint deberán ir completándose. Para poder establecer el valor de esta variable se requieren de las variables: auxPuntosPorSprint y

auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor de la velocidad ideal para dicho sprint. Unidad:puntos.

auxCalculaPuntos: esta variable auxiliar permite determinar la velocidad ideal con la cual los puntos establecidos para cada Sprint deberán ir completándose. Para poder establecer el valor de esta variable la misma requiere de las variables: auxPuntosPorSprint y auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor de la velocidad ideal para dicho sprint. Unidad: puntos.

PUNTOSPORSPRINT: es la primera variable de nivel del Subsistema, y en ella se almacenan los puntos que deberán ser desarrollados a lo largo de cada uno de los Sprints, y que provienen de la variable auxPuntos. Unidad: puntos.

BURDOWNCHARTS : esta variable de nivel representa lo que en la realidad es el BurdownChart(2.5). La misma muestra cómo deberían desarrollarse los puntos previstos para cada Sprint, para esto el descuento de puntos se realiza en la medida que lo indique la variable auxCalculaPuntosPorHacer, y de esta manera el nivel presentará una forma de picos y caídas Unidad: puntos.

PUNTOSPORHACER: variable de nivel que representa la cantidad de puntos que deberían ir desarrollándose a lo largo del Sprint y en condiciones normales de trabajo, para esto el incremento de puntos se realiza en la medida del valor que tome la variable auxCalculaPuntos. Al igual que el nivel anterior presentará una forma de picos. Unidad: puntos.

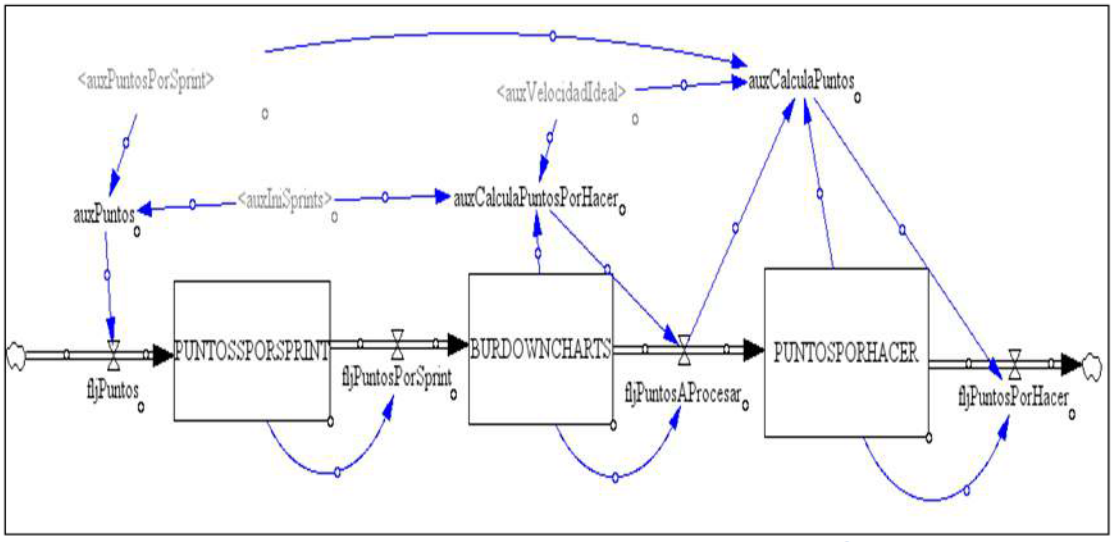


Figura 33 - Diagrama de Forrester Subsistema de Producción

5.4 Subsistema de Desarrollo de Tareas

En la Figura 34 se presenta el diagrama de Forrester correspondiente al este Subsistema de Desarrollo de Tareas, y en ella se pueden apreciar las variables detalladas a continuación.

tareasPorPuntos: esta variable permite establecer la cantidad de tareas por puntos de historia que el Scrum Master estime o considere oportunas dada la complejidad de las tareas a desarrollar. Unidad: tareas

cantidadTareasPorPuntos: representa la cantidad total de tareas que se deben realizar para poder completar cada uno de los puntos de historia establecidos por Sprint. El valor de esta variable surge del producto entre el valor asignado a tareasPorPuntos y auxPuntosPorSprint. Dicho producto se efectuará solamente si PUNTOSPORHACER es mayor a cero. Unidad: tareas.

TAREASPLANIFICADAS: variable de nivel que representa la cantidad total de tareas a desarrollar en los diferentes Sprints. El valor de esta lo determina cantidadTareasPorPuntos. Unidad: tareas.

TAREASPENDIENTESCODIF: este nivel representa la cantidad de tareas pendientes de codificación en cada uno de los Sprints. El valor inicial está determinado por las tareas originales que se planificaron, y se ve modificado de acuerdo a la cantidad de Tareas que pudieran surgir de las diferentes tareas con Errores y las Tareas Extras. Unidad: tareas,

TAREASCODIFICADAS: almacena la cantidad de tareas que se fueron codificando a lo largo del desarrollo del Sprint. El desarrollo presenta la forma general de un diente de sierra, con puntos altos y descensos lineales, que se ven alterados por la ausencia no planificada de los integrantes del Team o por la aparición de Tareas no planificadas. Unidad: tareas.

auxAusenciaProduccion: esta variable auxiliar permite establecer en cuanto disminuirá la velocidad normal de producción diaria en función de la cantidad de ausencias no acordadas de los integrantes, de cuantos integrantes no asistieron al Sprint diario y de la velocidad de trabajo estimada inicialmente. Unidad: puntos.

tasaAjusteVelocidad: variable que permite ajustar la velocidad ideal establecida en subsistema de la sección 5.4. Ya que el número de tareas en relación a los puntos de historia no siempre es una tarea a un punto, se hace necesario modificar la velocidad de codificación para que los tiempos establecidos sean respetados. Unidad: tasa.

TAREASCONERRORES: variable de Nivel que acumula la totalidad de las tareas con errores sean de Integración o de Codificación. Su valor surge de la suma de la cantidad de tareas con errores de codificación dividido la cantidad de pruebas, más la cantidad de tareas con errores de integración divido la cantidad de pruebas de integración. Unidad: tareas.

auxTotalErroresDesarrollo: variable auxiliar donde se calcula el número de tareas con errores. Su valor se establece a partir de la suma de dos valores, ambos surgidos de dividir, el total de pruebas con errores de integración sobre el número de pruebas de integración, y del total de errores por codificación detectados sobre el número de pruebas realizadas. Unidad: errores

auxTareasARecodificar: variable auxiliar que permite determinar el número total de tareas con errores existentes, generando picos que representan el número de tareas que surgen a lo largo del proyecto. Unidad: tareas.

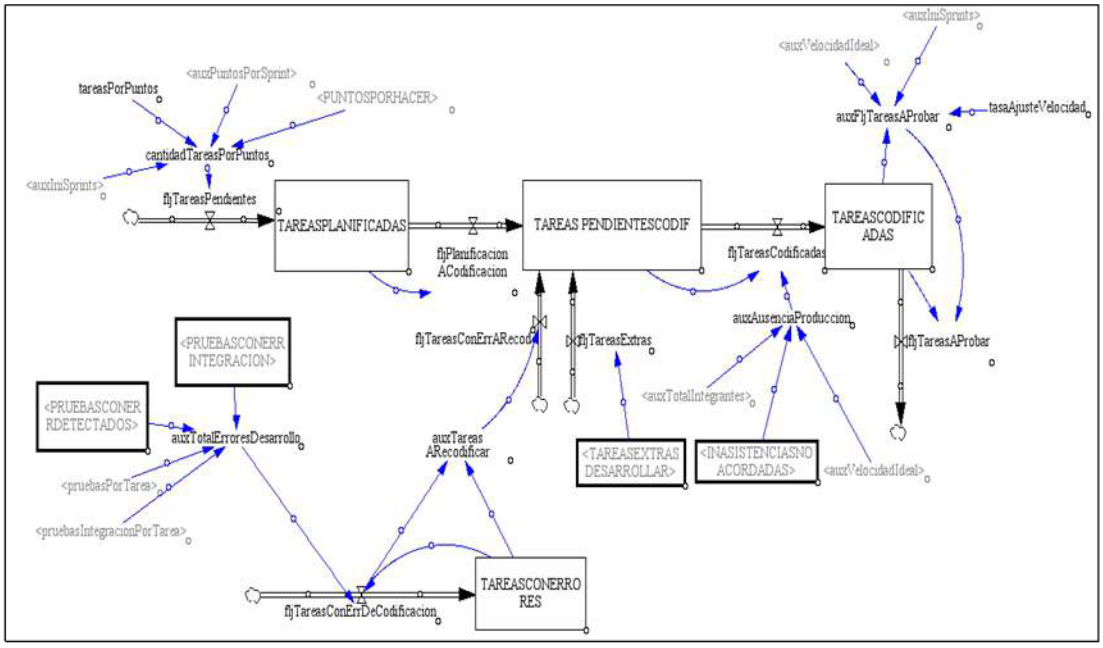


Figura 34 - Diagrama de Forrester Subsistema de Desarrollo de Tareas

5.5 Subsistema de Control de Errores de Desarrollo

En la Figura 35 se presenta el diagrama de Forrester correspondiente al este Subsistema de Control de errores de desarrollo, y en ella se pueden apreciar las variables detalladas a continuación.

pruebasPorTarea: esta variable auxiliar permite establecer la cantidad de pruebas que se realizarán sobre cada una de las distintas tareas que se codificaron en el Subsistema de la sección 5.3. Unidad: prueba

auxTotalPruebas: el valor de esta variable auxiliar resulta de multiplicar la cantidad de tareas programadas por el valor que contenga la variable pruebasPorTarea y sirve de entrada para el Subsistema. Unidad: prueba

PRUEBASADISENAR: esta variable de nivel almacena la totalidad de las pruebas que se determinen para cada tarea codificada en cada uno de los sprints. Unidad: prueba

PRUEBASAREALIZAR: a partir del nivel PRUEBASDISENADAS, esta variable almacena las pruebas diseñadas y que deberán realizarse para comprobar si las tareas pasan al proceso de integración o bien deben ser reprogramadas. Unidad: prueba

PRUEBASCONERRDETECTADOS: el contenido de esta variable de nivel representa aquellas tareas que fueron probadas y en las que se encontraron algún tipo de error. El error producido puede darse tanto por la presión en el plazo como por cansancio del equipo. Unidad: prueba.

PRUEBASSINERROR: esta variable de nivel representa aquellas tareas que fueron terminadas y en las cuales no fueron encontrados errores de ningún tipo. Su valor surge de la diferencia entre las tareas totales que fueron probadas y aquellas que tienen errores. Unidad: prueba.

tasaErrPorPresionPlazo: es una variable auxiliar que almacena un porcentaje o tasa de presión que sufre el Team y que se estima en el subsistema de la sección 5.7. El porcentaje de esta variable es aplicado sobre el flujo fljErroresPorCodificacion y permite regular el nivel de la variable de nivel PRUEBASCONERRDETECTADOS. Esta tasa se ve disminuida por la experiencia que demuestre el Team. Unidad: tasa.

tasaErrPorCansancio: es una variable auxiliar que almacena un porcentaje o tasa de cansancio que sufre el Team y que se estima en subsistema de la sección 5.12. El

porcentaje de esta variable es aplicado sobre el flujo fljErroresPorCodificacion y permite regular el nivel de la variable de nivel PRUEBASCONERRDETECTADOS. Unidad: tasa.

pruebasIntegracionPorTarea: variable que indica el número total de pruebas de integración que serán realizadas sobre aquellas tareas codificadas que no presenten errores. Unidad: prueba.

auxTotalPruebasIntegracion: variable auxiliar que representa el resultado de la multiplicación de la cantidad de pruebas de integración a realizar por el número de tareas. Esta variable permite vincular el actual Subsistemas con el Subsistema de Pruebas de Integración. Unidad: prueba.

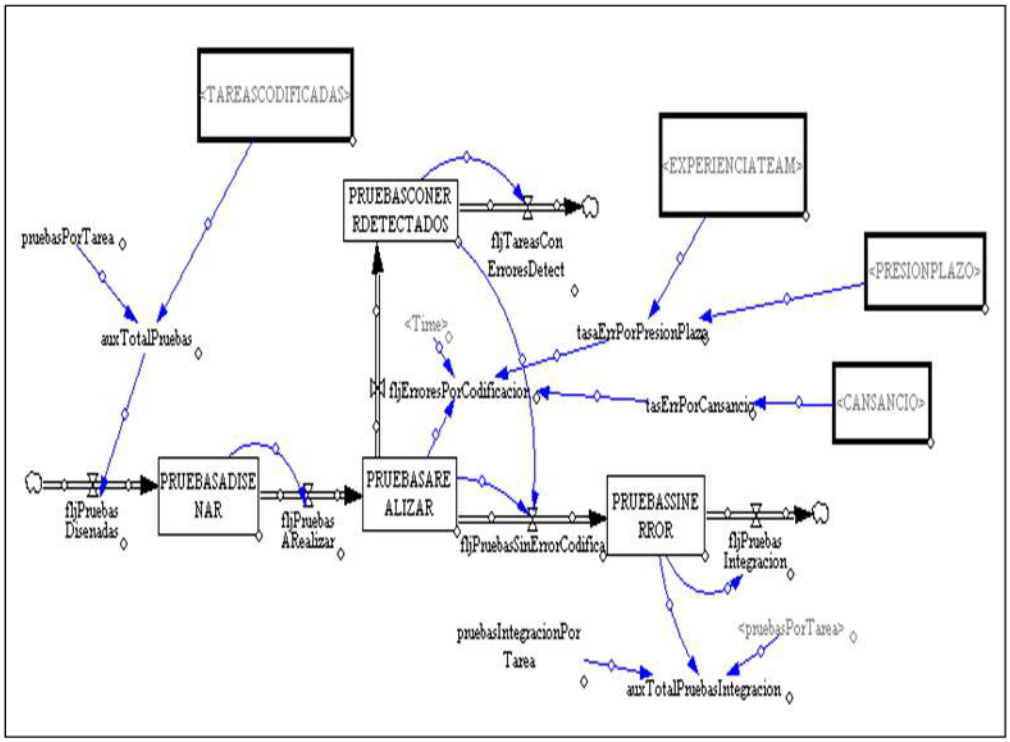


Figura 35 – Diagrama de Forrester Subsistema de Control de Errores de Desarrollo

5.6 Subsistema de Pruebas de Integración

En la Figura 36 se presenta el diagrama de Forrester correspondiente al este Subsistema de Control de errores de desarrollo, y en ella se pueden apreciar las variables detalladas a continuación.

PRUEBASINTEGRACIONDISENADAS: a partir del contenido de la variable auxTotalPruebasIntegracion, esta variable de nivel representa las pruebas de integración que se diseñaran y deberán realizarse sobre aquellas tareas que no presentaron errores de codificación y sobre las que se realizaran las pruebas de integración. Unidad: prueba.

PRUEBASINTEGRACIONREALIZADAS: las pruebas de integración diseñadas en el nivel anterior son llevadas a cabo, y esta variable representa esas pruebas una vez que fueron diseñadas. Unidad: prueba.

tasaPruebasErrIntegracion: es una variable auxiliar que permite establecer qué porcentaje de las pruebas de integración realizadas presentan o no errores de ese tipo. Unidad: tasa.

PRUEBASINTEGRADAS: a partir del valor establecido en tasaPruebasErrIntegracion y de las pruebas de integración realizadas esta variable de nivel toma su valor. En la

medida que la tasa disminuye el valor almacenado en esta variable aumenta. Unidad: prueba.

PRUEBASCONERRINTEGRACION: a partir del valor establecido en tasaPruebasErrIntegracion y de las pruebas de integración realizadas esta variable de nivel toma su valor. En la medida que la tasa aumente el valor almacenado en este nivel aumenta. Unidad: prueba.

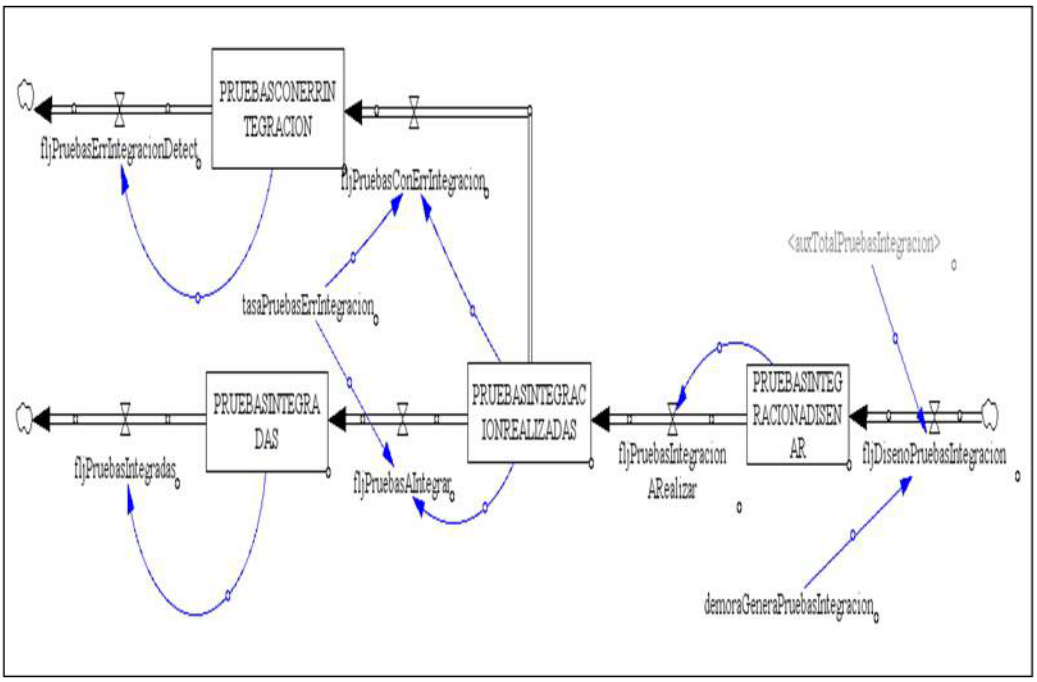


Figura 36 – Diagrama de Forrester Subsistema de Pruebas de Integración

5.7 Subsistema de Presión en el Plazo

En la Figura 37 se presenta el diagrama de Forrester correspondiente a este Subsistema, y en ella se pueden apreciar las variables detalladas a continuación.

lkpPresionPlazo: variable tipo *LookUp* que establece que presión tendrá el Team en la medida que el tiempo del proyecto transcurra. Tiene como parámetro de entrada el porcentaje de avance del tiempo, y como salida para cada entrada la tasa o porcentaje de presión que sufrirá el Team. Unidad: tasa.

auxFinSprintReal: variable auxiliar que calcula el porcentaje de tiempo transcurrido del proyecto. Se calcula como un cociente entre *FINAL TIME* y *Time*. Unidad: hora.

auxCoeficientePresionPlazo: el valor de esta variable auxiliar determina la presión que sufre el Team a partir de la tasa de presión que surja de entrecruzar los valores de auxFinSprintReal y la variable lkpPresionPlazo. Unidad: tasa.

diferenciaEnLaPresionPlazo: almacena la diferencia en la presión en el plazo en la medida que avanza el tiempo y la presión que ya viene sufriendo el Team. Unidad: tasa.

PRESIONPLAZO: es la variable de nivel que almacena la presión total que el Team sufre a lo largo del desarrollo del proyecto. Esta presión total solo es disminuida por la Experiencia que tenga el Team, ya que se asume, que a mayor experiencia la presión por plazo no se sufre de manera directa. Unidad: presión.

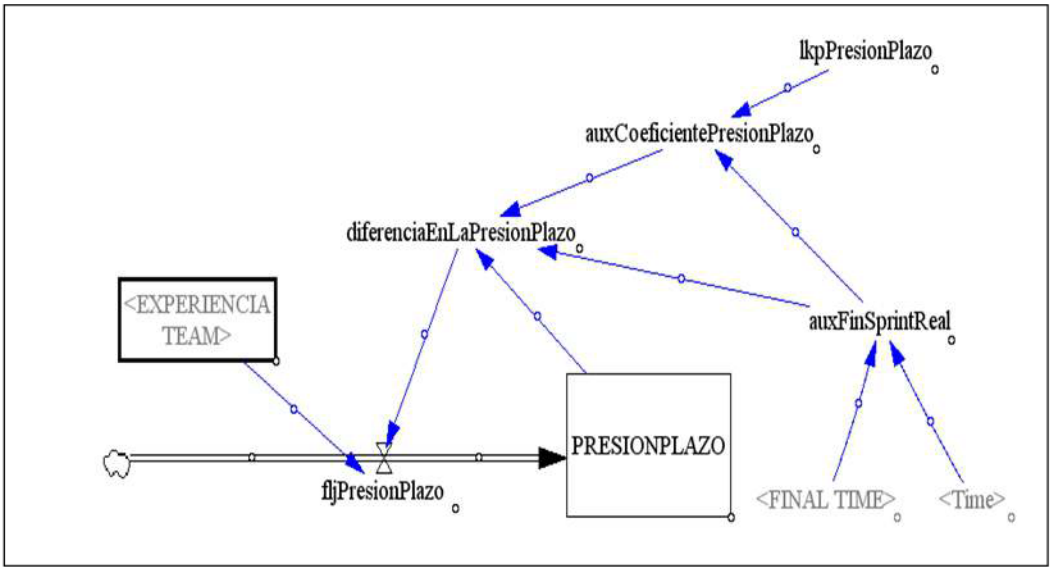


Figura 37– Diagrama de Forrester Subsistema de Presión en el Plazo

5.8 Subsistema Desarrollo Tareas Extras

En la Figura 38 se presenta el diagrama de Forrester correspondiente a este Subsistema, y en ella se pueden apreciar las variables detalladas a continuación.

probabilidadTareaExtraNoPlanificada: esta variable auxiliar almacena un valor que puede ser establecido en tiempo de corrida y que establece con que probabilidad una tarea extra no planificada podría aparecer a lo largo del proyecto. Unidad: tasa.

auxPeriodoTareaExtraNoPlanificada: el periodo en el cual podría aparecer una tarea extra no planificada es determinado en esta variable, y surge de la probabilidad de que una tarea extra aparezca y de la duración total del proyecto. Unidad: hora.

auxTareaExtraNoPlanificada: esta variable auxiliar genera una serie de pulsos unitarios mediante la implementación de la función "PULSE TRAIN". Donde cada pulso que se genera determina una tarea extra no planificada. Unidad: tarea.

lkpTareasExtrasSprint: es una variable tipo LookUp, donde el contenido de la entrada es el número de sprint que se esté desarrollando, y el valor de salida asociado a dicha entrada es la cantidad de tareas extras planificadas que se estiman podrían aparecer a lo largo del proyecto en dicho sprint. Unidad: tarea.

auxTareasExtrasPorHacer: variable auxiliar cuyo contenido surge del valor establecido en SPRINTSPLANIFICADOS y lkpTareasExtrasSprint, y determina cuantas tareas extras planificadas podrían llegar a implementarse en cada uno de los Sprints. El rol de auxIniSprints en esta variable es la de poder saber en qué momento se inicia el sprint y en función de si un sprint se inició se considera el valor de SPRINTSPLANIFICADOS, caso contrario se considera iniciado el sprint. Unidad: tarea.

TAREASEXTRASDESARROLLAR: variable de nivel que almacena las diferentes tareas extras, tanto planificadas como no planificadas y que deberán ser integradas al sistema de Desarrollo de Tareas (sección 5.4) Unidad: tarea.

TAREASEXTRASREALIZADAS: variable de nivel que almacena el número total de tareas extras realizadas a lo largo del proyecto. Unidad: tarea.

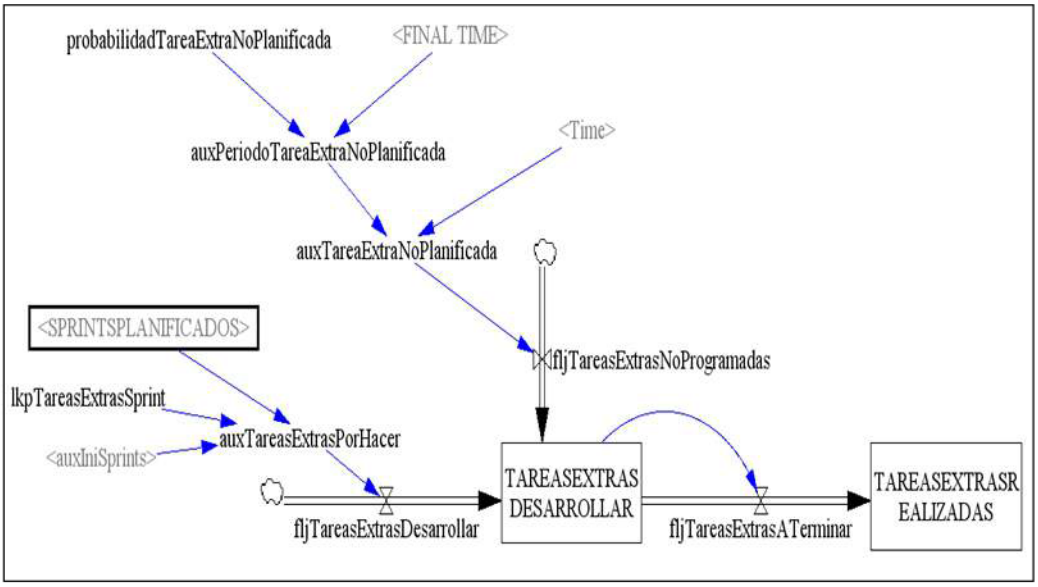


Figura 38– Diagrama de Forrester Subsistema de Desarrollo de Tareas Extras

5.9 Subsistema de Búsqueda y Reclutamiento

Las variables descriptas en el presente apartado se pueden apreciar en el diagrama de Forrester de la Figura 39 el cual corresponde al presente Subsistema.

nroIdealIntegrantes: variable auxiliar que estable el número de integrantes considerado ideal u óptimo para el proyecto. El número puede ser establecido por el operador siendo un número variable entre 4 y 8. Unidad: persona.

auxTotalIntegrantes: esta variable almacena el número total de los integrantes del Team, y surge de la suma de la cantidad de integrantes que existieran en cada uno de los niveles de integrantes considerados para el presente trabajo. Unidad: persona.

diferenciaIntegrantesTeam: es la diferencia de integrantes existente entre el número ideal y la cantidad real actual de los integrantes, y es utilizada para saber cuántos integrantes deberían contratarse para completar el número ideal de integrantes. Unidad: persona.

auxNroCandidatos: es un número generado al azar mediante la función "RANDOM UNIFORM" provista por la herramienta VenSim. Unidad: persona.

RHPRESELECCIONADOS: esta variable de nivel almacena la cantidad de candidatos que fueron preseleccionados para ser contratados. La cantidad de candidatos surge de un número aleatorio entre la cantidad de candidatos y la diferencia de integrantes en el Team. Unidad: persona.

porcentajeSeniorsBuscados: esta variable auxiliar permite al operador seleccionar el porcentaje de candidatos Seniors buscados para completar la plantilla de integrantes del Team, o bien para seleccionar la cantidad de Juniors buscados. Unidad: tasa.

RHCANDIDATOSSENIORS: el valor almacenado en esta variable de nivel surge del valor asignado a la variable porcentajeSeniorsBuscados y que se aplica sobre la cantidad de personas seleccionadas representadas en el nivel RHPRESELECCIONADOS. Unidad: persona.

RHCANDIDATOSJUNIORS: el valor que toma resulta de la diferencia entre 1 y el porcentaje de Seniors buscados, así esta variable de nivel almacena aquellos candidatos pre-seleccionados que no son Seniors. Unidad: persona.

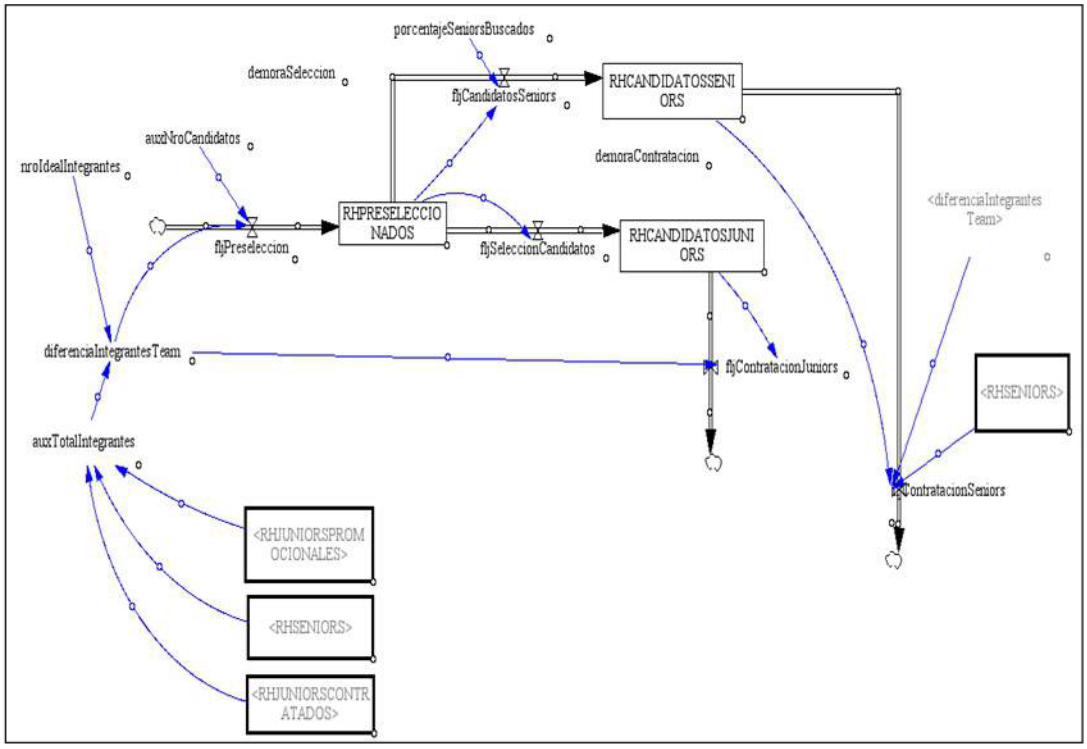


Figura 39 - Diagrama de Forrester Subsistema de Búsqueda y Reclutamiento

5.10 Subsistema de Promoción

Las variables descriptas en el presente apartado se pueden apreciar en el diagrama de Forrester de la Figura 40 el cual corresponde al presente Subsistema.

RHJUNIORSCONTRATADOS: esta variable de nivel representa al número de aquellos candidatos Juniors que fueron seleccionados y que finalmente fueron contratados. Unidad: persona.

tiempoSalidaJuniors: variable auxiliar que permite administrar al operador un valor que establece el periodo de tiempo en el cuál se producirán abandonos del Team por parte de los Juniors contratados. De manera resumida se puede decir que funciona como un regulador del abandono periódico de integrantes noveles. Unidad: hora

tasaAbandonoJuniors: es una variable auxiliar cuyo valor puede ser seleccionado por el operador y determina qué porcentaje de los integrantes Juniors contratados dejaran el Team. Funciona de manera conjunta con la variable tiempoSalidaJuniors, así estableciendo cierto periodo de tiempo un cierto número de integrantes dejará el Team. Unidad: tasa.

cantidadPasan: es una variable que determina que número de integrantes Juniors contratados pasará al siguiente nivel. En este caso particular no es una tasa, si no, un número fijo que el operador puede seleccionar para el pase. Unidad: persona.

demoraPaseAJuniors: variable auxiliar por la cual se puede determinar el tiempo mínimo que los integrantes serán considerados Juniors antes de ser pasados al nivel de promoción. Unidad: hora.

RHJUNIORSPROMOCIONALES: es variable de nivel representa un estadio intermedio entre los integrantes Juniors contratados y los Seniors. Unidad: persona.

tiempoSalidaPromocionales: variable auxiliar que permite administrar al operador un valor que establece el periodo de tiempo en el cuál se producirán abandonos del Team por parte de los Juniors promocionales. De manera resumida se puede decir que

funciona como un regulador del abandono periódico de integrantes noveles. Unidad: hora.

tasaAbandonoPromocionables: es una variable auxiliar cuyo valor puede ser seleccionado por el operador y determina qué porcentaje de los integrantes Juniors Promocionales dejaran el Team. Funciona de manera conjunta con la variable tiempoSalidaPromocionalesestableciendo cierto periodo de tiempo un cierto número de integrantes dejará el Team. Unidad: tasa.

tasaAvanceASenior: tasa seleccionada por el operador y que determina el porcentaje de los Juniors promocionales que podrían pasar a ser Seniors de aquellos que aún se encuentran en el nivel de RHSENIORSPROMOCIONALES. Unidad: tasa.

RHSENIORS: variable de nivel que almacena la cantidad de integrantes considerados Seniors dentro del Team. El nivel puede conformarse tanto por personas que son contratadas de manera directa, mediante las promociones o bien integrantes que ya conforman el Team inicialmente. Unidad: persona.

tiempoSalidaSeniors: variable auxiliar que permite administrar al operador un valor que establece el periodo de tiempo en el cuál se producirán abandonos del Team por parte de los Seniors. De manera resumida se puede decir que funciona como un regulador del abandono periódico de integrantes Seniors. Unidad: hora.

tasaAbandonoSeniors: es una variable auxiliar cuyo valor puede ser seleccionado por el operador y determina qué porcentaje de los integrantes Seniors dejarán el Team. Funciona de manera conjunta con la variable tiempoSalidaSeniors, estableciendo que en un cierto periodo de tiempo un cierto número de integrantes dejará el Team. Unidad: hora.

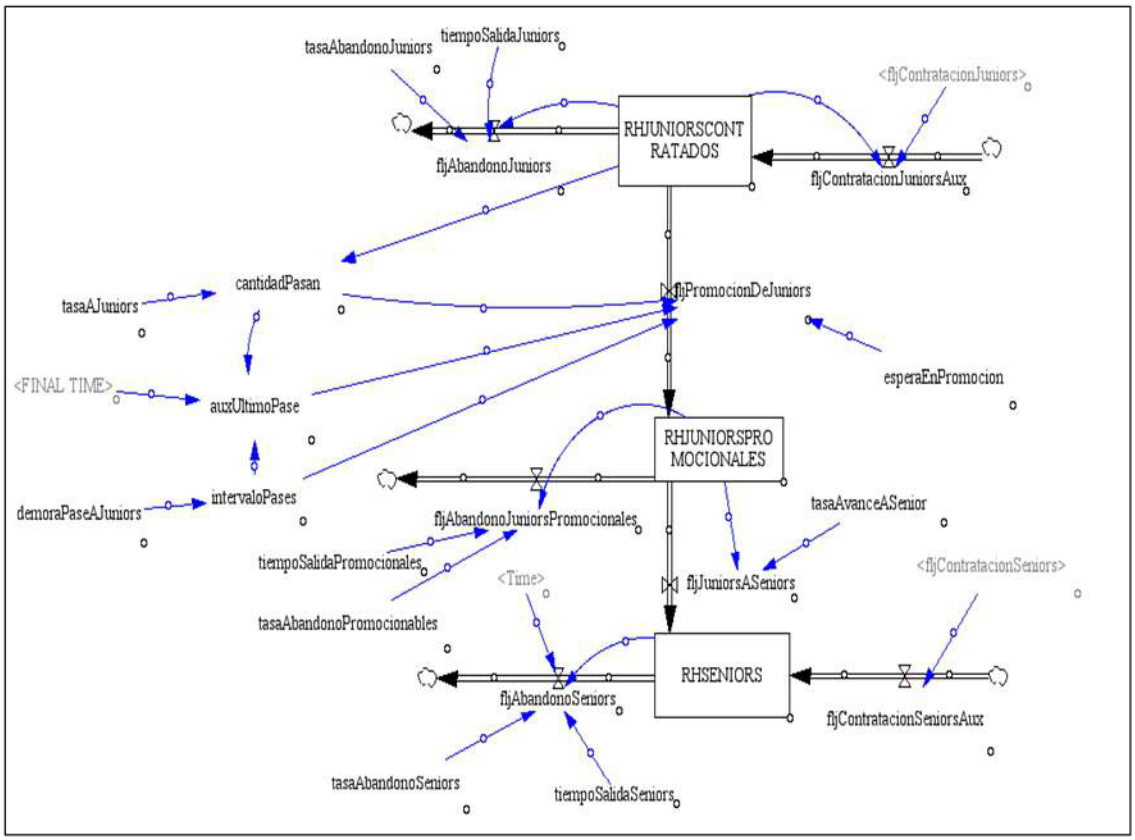


Figura 40 - Diagrama de Forrester Subsistemas de Promoción

5.11 Subsistema de Adquisición de Experiencia

Las variables descriptas en el presente apartado se pueden apreciar en el diagrama de Forrester de la Figura 41 el cual corresponde al presente Subsistema.

tasaAprendizajeJrsContratados: variable auxiliar que establece el valor para la tasa de aprendizaje de los Juniors Contratados. En la medida que el valor de esta variable aumente la rapidez de la adquisición de experiencia será mayor por parte de los integrantes del Team. Unidad: tasa.

auxNivelProductividadJuniors: variable que permite al Scrum Master establecer en que porcentaje la experiencia de los integrantes Juniors inciden en el nivel de aprendizaje. Unidad: tasa.

auxPonderacionJuniors: variable auxiliar que permite regular el peso que tendrán los Juniors al momento de ponderar la experiencia total de todos los integrantes del Team. Esta función es logarítmica y está dada por la ecuación: $1- e^{(-tasaAprendizajeSeniors*Time)}$ [39]. Unidad: tasa

tasaAprendizajePromocionales: variable auxiliar que establece el valor para la tasa de aprendizaje de los Juniors Promocionales. En la medida que el valor de esta variable aumente la rapidez de la adquisición de experiencia será mayor por parte de los integrantes del Team. Unidad:tasa.

auxNivelProductividadJuniorsPromos: variable que permite al Scrum Master establecer en que porcentaje la experiencia de los integrantes Juniors en condición de ser promocionados inciden en el nivel de aprendizaje. Unidad: tasa.

auxPonderacionPromocionales: variable auxiliar que permite regular el peso que tendrán los Juniors Promocionales al momento de ponderar la experiencia total de todos los integrantes del Team. Esta función es logarítmica y está dada por la ecuación: $1- e^{(-tasaAprendizajeSeniors*Time)}$ [39]Únidad: tasa.

auxExperienciaJuniorsPromocionales: variable auxiliar que almacena el resultado de calcular la experiencia de los Juniors Contratados. Unidad: experiencia.

tasaAprendizajeSeniors: variable auxiliar que establece el valor para la tasa de aprendizaje de los Seniors. En la medida que el valor de esta variable aumente la rapidez de la adquisición de experiencia será mayor por parte de los integrantes del Team. Unidad: tasa.

auxNivelProductividadSeniors: variable que permite al Scrum Master establecer en que porcentaje la experiencia de los integrantes Seniors inciden en el nivel de aprendizaje. Unidad: tasa.

auxPonderacionSeniors: variable auxiliar que permite regular el peso que tendrán los Seniors al momento de ponderar la experiencia total de todos los integrantes del Team. Esta función es logarítmica y está dada por la ecuación: $1- e^{(-tasaAprendizajeSeniors*Time)}$ [39] Unidad: tasa.

auxExperienciaSeniors: variable auxiliar que almacena el resultado de calcular la experiencia de los Seniors. Unidad: experiencia.

EXPERIENCIATEAM: esta variable de nivel acumula y representa el nivel de experiencia que presente el Team. A partir de la ponderación que se les asigne a los integrantes del Team esta variable presentará un comportamiento de rápido crecimiento o será más lento dicho crecimiento. Unidad:experiencia.

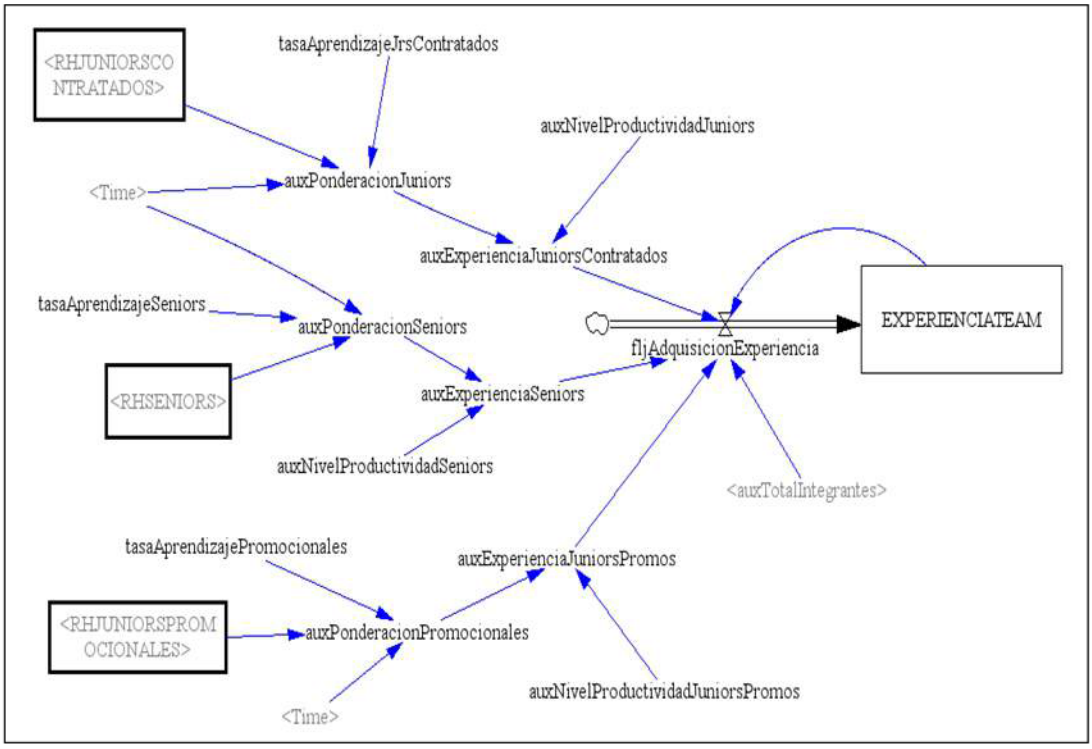


Figura 41 – Diagrama de Forrester Subsistema de Adquisición de Experiencia

5.12 Subsistema de Cansancio

Las variables descriptas en el presente apartado se pueden apreciar en el diagrama de Forrester de la Figura 42 el cual corresponde al presente Subsistema.

lkpCoeficienteCansancioDiario: variable tipo LookUp que establece una tasa de cansancio en función de la cantidad de horas trabajadas. Así cuanto más horas diarias se trabajan mayor es la tasa de cansancio asignada. La variable tiene como entrada el número de horas diarias normales planificadas para cada Sprint, y como salida la tasa de cansancio, estimada por la experiencia del Scrum Master, para dicha cantidad de horas. Unidad: tasa.

auxTasaCansancioDiario: en esta variable se determina la tasa de cansancio para la cantidad total de horas trabajadas por día. El valor se determina a partir de la suma de las horas diarias establecidas en axuHorasDiarias y de las horas extras trabajadas en la jornada establecida en TAREASEXTRASDESARROLLAR. Así la variable LookUp retornará la tasa correspondiente para la entrada que resulte de la cantidad de horas trabajadas. Unidad: tasa

auxFinSprintCansancio: variable auxiliar donde se calcula el porcentaje de tiempo del proyecto transcurrido, a partir del valor de "Time" sobre "FINAL TIME". Unidad: hora.

lkpCoeficienteCansancio: variable tipo LookUp que establece una tasa de cansancio en función del tiempo trabajado por parte del Team. Así cuanto mayor sea el tiempo de trabajo en el proyecto mayor es la tasa de cansancio asignada. La variable tiene como entrada el número de horas trabajadas en el proyecto, y como salida la tasa de cansancio, la cual es estimada por la experiencia del Scrum Master. Unidad: tasa.

auxTasaCansancio: en esta variable auxiliar se establece la tasa de cansancio estimada para el Team en la medida que transcurre el tiempo y el proyecto. A partir del cociente entre el nivel de horas trabajadas (sección 5.10), y el tiempo se determina la entrada para la variable LookUp, y con esto se obtiene la tasa correspondiente que le fuera asignada al inicio del proyecto. Unidad: tasa,

diferenciaCansancio: esta variable auxiliar promedia las tasas de cansancio, y realiza la resta correspondiente de la variable de nivel CANSANCIO. De esta manera la variable cansancio logra mantener el nivel de cansancio total del Team. Unidad: tasa.

CANSANCIO: variable que almacena y representa el total del cansancio que presenta el Team a lo largo del proyecto. Presenta un comportamiento ascendente, y este ascenso será más o menos pronunciado en función de las tasas que se asignen a las variables auxiliares Unidad: cansancio.

lkpCoeficienteCansancioDiario: variable tipo LookUp que establece una tasa de cansancio en función de la cantidad de horas trabajadas. Así cuanto más horas diarias se trabajan mayor es la tasa de cansancio asignada. La variable tiene como entrada el número de horas diarias normales planificadas para cada Sprint, y como salida la tasa de cansancio, estimada por la experiencia del Scrum Master, para dicha cantidad de horas. Unidad: tasa.

auxFinSprintCansancio: variable auxiliar donde se calcula el porcentaje de tiempo del proyecto transcurrido, a partir del valor de "Time" sobre "FINAL TIME". Unidad: hora

auxTasaCansancio: en esta variable auxiliar se establece la tasa de cansancio estimada para el Team en la medida que transcurre el tiempo y el proyecto. A partir del cociente entre el nivel de horas trabajadas (sección 5.10), y el tiempo se determina la entrada para la variable LookUp, y con esto se obtiene la tasa correspondiente que le fuera asignada al inicio del proyecto. Unidad: tasa,

diferenciaCansancio: esta variable auxiliar promedia las tasas de cansancio, y realiza la resta correspondiente de la variable de nivel CANSANCIO. De esta manera la variable cansancio logra mantener el nivel de cansancio total del Team. Unidad: cansancio

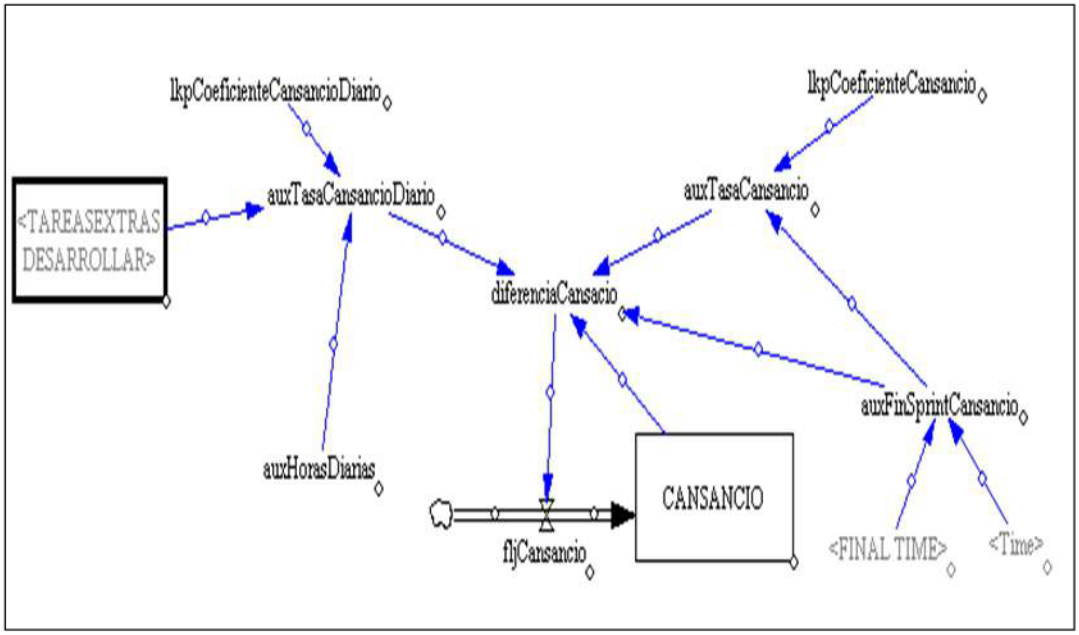


Figura 42 – Diagrama de Forrester Subsistema de Cansancio

5.13 Subsistema de Horas Trabajadas

Las variables descriptas en el presente apartado corresponden al presente Subsistema de Horas Trabajadas, cuyo diagrama de Forrester se puede apreciar en la Figura 43.

lkpSprintHorasExtras: variable tipo LookUp que establece la cantidad de horas extras planificadas por Sprints. Esta variable tiene como entrada el número de Sprint y

como salida la cantidad de horas extras asignadas previamente a cada Sprint. Unidad: hora.

auxHsExtrasTrabajadas: esta variable auxiliar almacena la cantidad horas extras planificadas menos las horas extras no acordadas (sección 5.6) que pudieran surgir en el sprint. La cantidad de horas extras planificadas surge de entrecruzar la variable auxIniSprint, la variable "Time" y el nivel SPRINTSPLANIFICADOS. De esta manera si el valor de auxIniSprint (sección 5.2) coincide con Time se obtiene el valor de entrada de la variable LookUp correspondiente al número de Sprint, y con esto la cantidad de horas extras planificadas. Unidad: hora.

lkpSprintDuracionEnHoras: variable tipo LookUp donde se establece la cantidad de horas normales planificadas para cada Sprints. Esta variable tiene como entrada el número de Sprint y como salida la cantidad de horas normales asignadas previamente a cada Sprint. Unidad: hora.

auxHorasNormalesTrabajadas: esta variable auxiliar almacena la cantidad horas normales planificadas para cada Sprint. La cantidad de horas normales planificadas surge de entrecruzar la variable auxIniSprint, la variable "Time" y el nivel SPRINTSPLANIFICADOS. De esta manera si el valor de auxIniSprint (seción 5.2) coincide con Time se obtiene el valor de entrada de la variable LookUp correspondiente al número de Sprint, y con esto la cantidad de horas normales planificadas. Unidad:hora.

HORASTRABAJADASPORSPRINT: variable de nivel que almacena las diferentes horas normales y horas extras trabajadas por el Team. SE presenta como una serie de picos que expresan las horas normales trabajadas y de las horas extras trabajadas. Permitiendo ver las mismas según el momento en el tiempo donde ocurrieron o se planificaron. Unidad: hora.

HORASTOTALESTRABAJADAS: variable de nivel que almacena la cantidad total de horas trabajadas por el Team. En condiciones normales presenta un crecimiento escalonado, donde se van acumulando las horas totales al momento de cada Sprint. Unidad: hora.

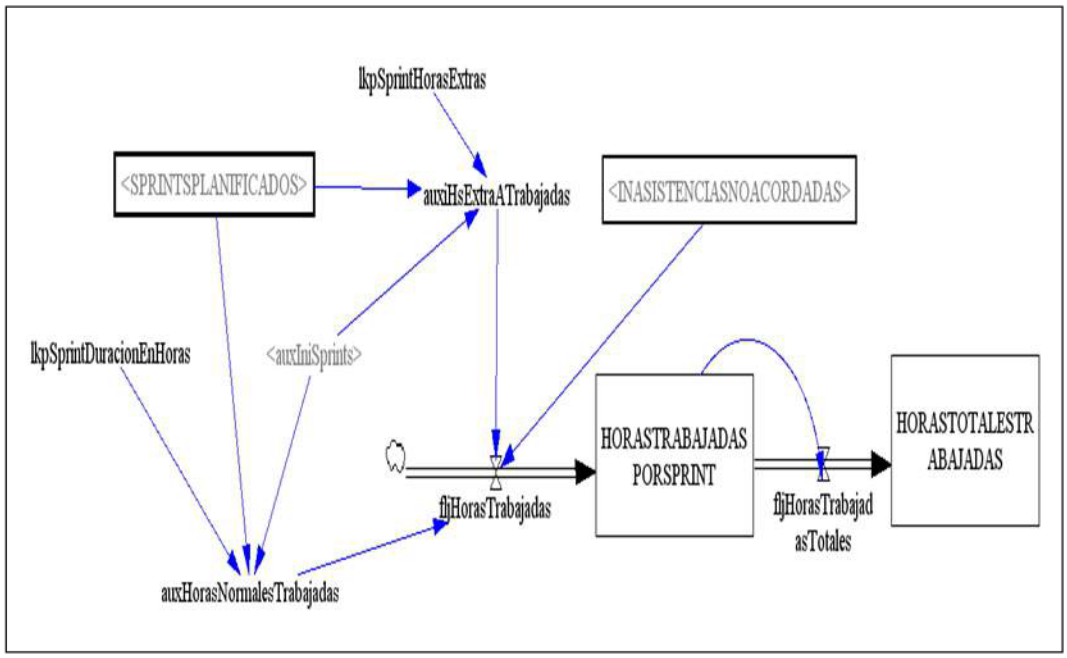


Figura 43 – Diagrama de Forrester Subsistema de Horas Trabajadas

5.14 Subsistema de Inasistencias

En el presente punto se detallan las variables correspondientes al Subsistema de Inasistencias, cuyo diagrama se aprecia en la Figura 44.

probabilidadInasistenciaNoAcordada: variable auxiliar donde se establece con que probabilidad puede aparecer una inasistencia no acordada durante el desarrollo del proyecto. Unidad: tasa.

auxPeriodoAsistenciaNoAcordada: esta variable auxiliar en forma conjunta con probabilidadInasistenciaNoAcordada y la variable "FINAL TIME" genera un valor que es un periodo de tiempo en el cual podría aparecer una inasistencia no acordada. Unidad: hora.

auxInasistenciasNoAcordadas: esta variable auxiliar representa las diferentes inasistencias que pudieran surgir en intervalos de tiempo calculado en auxPeriodoAsistenciaNoAcordada. Mediante la función "PULSE TRAIN" genera una seria de pulsos en los intervalos antes mencionados. El valor del pulso tiene una altura que dependerá a su vez, de la cantidad de integrantes ausentes ese día. Unidad: inasistencia.

auxIntegrantesAusentes: esta variable auxiliar genera de manera aleatoria un número de integrantes que varía de 1 al máximo de integrantes del Team. El valor que asuma influirá sobre la variable auxInasistenciasNoAcordadas. Unidad: persona.

INASISTENCIASNOACORDADAS: variable de nivel que representa las diferentes inasistencias y el momento donde estas ocurrieron a lo largo del tiempo de desarrollo del proyecto. Presenta la forma de picos, donde la altura indica la cantidad total de inasistencias no acordadas que hayan ocurrido en un momento dado. Unidad: inasistencia,

TOTALINASISTENCIASNOACORDADAS: esta variable de nivel almacena la cantidad total de inasistencias no acordadas que ocurrieron a lo largo del proyecto. Unidad: inasistencia.

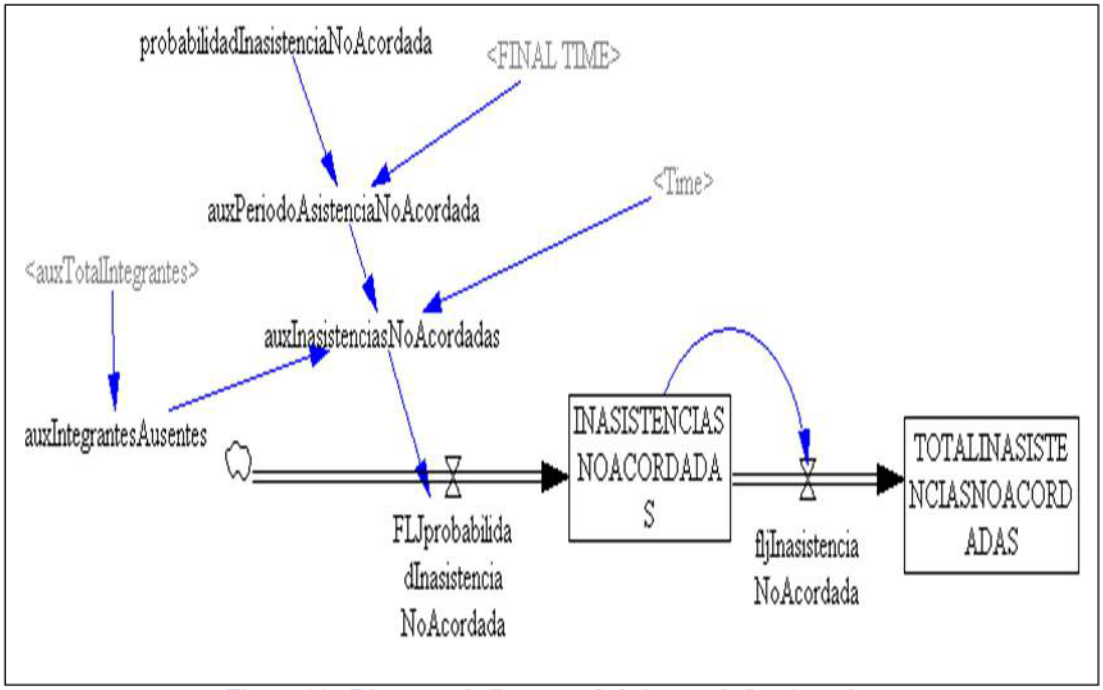


Figura 44 – Diagrama de Forrester Subsistema de Inasistencias

5.15 Limitaciones del Modelo de Simulación

En esta sección se expondrán las diferentes restricciones que el modelo desarrollado presenta tanto en su comportamiento, como aquellas que surgieron al momento de su implementación.

Es válido aclarar que el modelo de simulación presentado en este trabajo pretende simular un proyecto de desarrollo de software desarrollado bajo la metodología Scrum a partir del inicio y hasta el final de cada uno de los diferentes Sprints que se planifiquen para el proyecto.

Una de las restricciones que presenta el modelo es que no se consideran las reuniones previas al inicio de los Sprints donde el Scrum Master y el dueño del producto acuerdan las historias de usuario, y definen las prioridades de dichas historias. Tampoco se considera el momento donde el Team hace las estimaciones de tiempos o de velocidad para las historias de usuario seleccionada, por todo esto, dichas actividades se asumen realizadas y finalizadas al momento de establecer los diferentes parámetros que permiten iniciar las diferentes simulaciones.

En cuanto a las actividades de Scrum Diario, Revisión del Sprint y Retrospectiva del Sprint deben ser consideradas dentro de las tareas a realizar en cada uno de los Sprint, por lo tanto es primordial que en los cálculos de tiempos de duración del Sprint vistos en el Subsistema de Planificación (sección 5.2) las duraciones de estas actividades estén incluidas.

En la sección 2.4.1 del presente trabajo se enumeran los diferentes roles que existen en Scrum, dichos roles al momento de la simulación no son considerados por que como también se menciona en dicho apartado Scrum es multidisciplinar y los miembros del equipo participan en todo lo que sea posible desarrollando múltiples roles dentro del Team. Por lo tanto a los fines de la simulación en base al presente modelo se considera poco trascendente incluir estos roles y el cambio de roles entre los miembros ya que solo aportaría un mayor número de variables a administrar y manejar.

Basado en [38] se consideró oportuno incluir la sección 4.10 donde se presentó el Subsistema de Recursos Humanos 3 (tres) niveles de experiencia diferentes, pero relacionados al conocimiento y trabajos previos con Scrum que hayan realizado los miembros del Team. Como el presente trabajo no hace referencia a ningún lenguaje de programación, framework de desarrollo de software, herramientas CASE o cuestiones vinculadas a estas, se asume que la experiencia en cualquier otra especialidad de los integrantes del Team está probada y no se consideró relevante modelarla.

Otra de las consideración es que entre las Tareas extras no se incluyen tareas que no estén relacionadas directamente al desarrollo del producto como tal, es por esto que actividades como reinstalación de servers, de sistemas operativos, etcétera quedan fuera de este concepto. Pero como si se mencionó en la sección 5.8 se consideran aquellas tareas que pudieran surgir por errores, omisiones, o fallas al momento del análisis, relevamiento de los requerimientos, o bien que pudieran surgir de errores al momento de la división de requerimientos en tareas a desarrollar.

En cuanto a la representación de los valores obtenidos de los diferentes cálculos realizados se mantuvieron tal cual, no se efectuaron redondeos.

Dada la amplitud de situaciones o casos que podrían poner en riesgo un proyecto, en el presente trabajo se consideran algunas situaciones que pueden recaer y según lo propuesto por Somerville en [25], en los considerados Riesgos del Proyecto y Riesgos del Producto. Así situaciones de riesgo como: Imposibilidad de Reclutar personal capacitado para el proyecto, personal clave ausente por causas diversas, el abandono de personal con experiencia, errores de comunicación que derivan en errores de diseño e integración del software desarrollado, sobreestimación de la capacidad de los integrantes, serán los considerados en el presente trabajo. Vale aclarar que no se realiza la clasificación de los mismos en cuanto a la magnitud del impacto o la relevancia del mismo.

Tampoco dentro del modelo se presentan o modelan problemas por comunicación en el Team, pero este tipo de problemas podría modelarse a partir del comportamiento de la variable que representa las tareas no planificadas.

En relación a la búsqueda de integrantes dada una diferencia entre el número de integrantes reales e ideales siempre se buscará completar el número ideal de integrantes, ya sea por los considerados Seniors, solo Juniors, o bien por un conjunto mixto de estos.

A modo de aclaración, en cuanto al comportamiento de las variables que presentan picos y descensos al comenzar/terminar los sprints planificados se debe a que se ha modelado teniendo al Spring como bloque de tiempo. Se ha tomado la decisión de modelarlo de esta manera por considerar que el Sprint presenta una granularidad intermedia en bloques de tiempo. Por ejemplo, en este trabajo el burdownchart muestra los Sprints de todo un proyecto y no un project burndownchart o day burndonchart. Es por ello que producen picos al inicio de los Sprint decayendo al final de los mismos.

En cuanto a las tareas extras el sistema modelado permite que el scrum master ingrese tareas extras, las cuales en base a su experiencia estima que podrían surgir durante el proyecto. En este sentido el scrum master podría realizar nuevas corridas previendo posibles tareas extras, como para conocer el peor escenario o cuando efectivamente ocurran las tareas para conocer como impactara en el proyecto.

6 Evaluación del Modelo de Simulación Construido

En este capítulo se presentan los casos de validación y pruebas realizadas con el modelo construido, que corresponde a la Fase de Evaluación de la metodología de Dinámica de Sistemas.

Primeramente se describen los 3 casos de validación que utilizan Scrum como metodología de desarrollo, y que por lo tanto se ajustan a los requerimientos buscados para la validación del modelo presentado en este trabajo.

Se presentan de manera conjunta los datos más relevantes de cada uno de los casos seleccionados, las tablas correspondientes a los parámetros que permiten iniciar cada corrida, y se incluyen también las Figuras obtenidas en cada corrida de simulación.

En cada una de las tablas se establece el Número de Sprint, la cantidad de Requerimientos que se completará en cada Sprint, la velocidad Ideal, la duración en días y la Duración en Horas de cada Sprint, y en la última columna el instante de inicio.

La columna Duración en Horas resulta del producto entre la cantidad de horas diarias a trabajar y los días que durará el Sprint.

La columna Velocidad Estimada resulta de la división entre la cantidad de Requerimientos por Sprints y la Duración en horas del Sprint correspondiente. La columna Horas hombres se calcula según [40].

Los Figuras mostradas en cada corrida representan a las variables más significativas para esta sección y permiten observar el comportamiento de modelo frente a cada escenario.

En este capítulo también se presentan tres casos experimentales de estudio de situaciones que pueden ocurrir durante del desarrollo de un proyecto gestionado con Scrum. Los casos van de menor a mayor en complejidad, y cuando más complejos son más variables intervienen. En cada caso se definirán políticas alternativas que permitan a los Scrum máster y al Team observar cual es la mejor para el caso concreto.

6.1 Casos de Validación

La validación se ha realizado siguiendo una combinación de los enfoques presentados en [41]. Como criterio Objetivo de Validación, se ha utilizado la correspondencia entre estructura y comportamiento, en este caso las situaciones reales que ocurren en los proyectos con los bucles de retroalimentación y los arquetipos sistémicos. Además de evidencia empírica de varios proyectos.

Por otro lado también en [41] se hace referencia a la validación por criterio de utilidad, en donde se especifica que si el modelo fue desarrollado para crear escenarios alternativos y tomar decisiones, entonces la utilidad depende de si el modelo logra construir el escenario adecuado, si puede establecerse que el escenario es realista (como opuesto a fantasioso) y si permite tomar decisiones informadas.

Es por ello que en este trabajo se tomaron los criterios mencionados para la validación.

6.1.1 Caso 1: "Automatización de sistemas de Desarrollo ágil"

El proyecto se desarrolló en el marco convenio entre la Universidad Autónoma de Barcelona y la empresa UNIT4. Esta empresa tiene como bandera las llamadas soluciones Ekon (actualmente UNIT4 Ekon) [42].

Como el contrato con la empresa especifica que el trabajo por parte de los becarios es a media jornada, una velocidad del 50% sería equiparable a una velocidad de 100% en un equipo a jornada completa.

6.1.1.1 Datos Principales del Proyecto

En la Tabla 3 se expresan los valores utilizados para la simulación realizada para la validación en el caso 1.

Tabla 3 – Caso de Validación 1 – Parámetros Iniciales

Sprint	Requerimientos	Puntos Historia	Puntos por Tareas Extras	Total Puntos	Duración en Horas	Velocidad Ideal Estimada	Inicio
1	9	18	0	18	70	0,268	0
2	8	28	0	28	60	0,350	70
3	12	31	0	31	80	0,387	130
4	9	21	0	21	57	0,262	210
5	9	24	0	24	60	0,300	267
6	7	25	0	25	56	0,312	327
7	10	25	0	25	63	0,312	383
Total	**64**	**172**	**0**	**172**	**446**	**-**	**446**

6.1.1.2 Otros Datos

- Integrantes: 6-Personas. 1-Scrum Master. 5-Team.
- Experiencia: Juniors.
- Número de Sprints: 8 de los cuales 1 fue para Release Planning Meeting (no incluido).
- Duración Total del Proyecto: 595 Horas, de las cuales 446 Horas son de Scrum.
- Horas Diarias: 8 horas.
- Horas Hombre Ideal: 8 horas por 1 punto historia
- Requerimientos: 1 por cada punto de historia.
- Pruebas por Requerimientos: 1 por cada Requerimiento.
- Horas Extras: No
- Errores por Presión en el plazo: No.
- Errores de Integración: No.
- Promociones en el Team: No.
- Abandonos en el Team: No.
- Tareas Extras: No.

6.1.1.3 Resultado de la Corrida

Los momentos en el tiempo para el inicio de cada Sprint son esenciales en el modelo ya que de esto dependen los valores que otras variables pueden asumir. En la Figura 45 se observa el inicio de cada uno de los sprints.

Los valores representados mediante la variable auxInicioSprint se aprecian en la siguiente Figura 45.

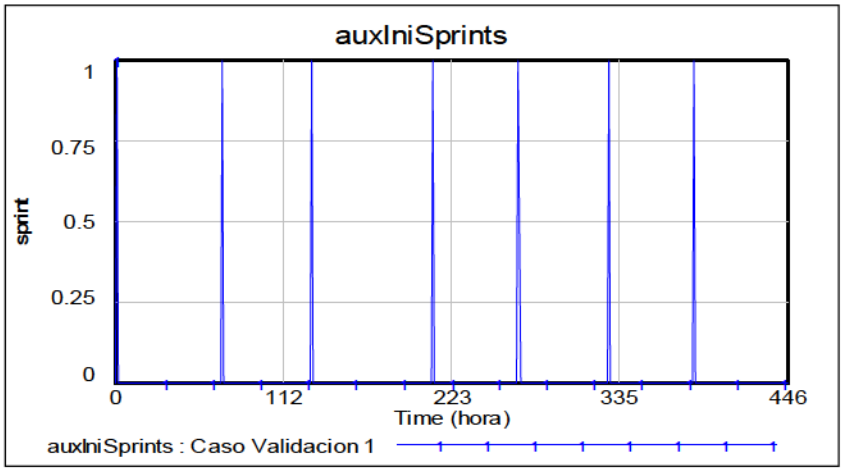


Figura 45 - Caso de Validación 1- Inicio Sprint

La cantidad de puntos de Historia seleccionados para cada Sprint establecidos en la variable lkpPuntosPorSprint y que se reflejan en la variable auxPuntosPorSprint se presentan en la Figura 46.

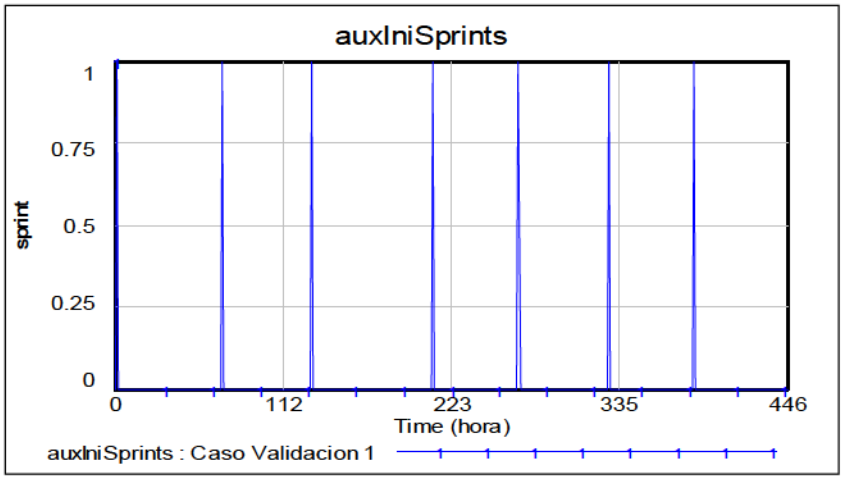


Figura 46 – Caso de Validación 1- Puntos Por Sprint

Como se mencionó anteriormente la variable auxPuntosPorSprint indica la cantidad de puntos a desarrollar en cada uno de los Sprints. Para poder llegar al final de cada Sprint con la totalidad de los puntos completados, el Team debe realizar un conjunto de tareas a una determinada velocidad.

De manera conjunta los puntos establecidos al inicio del sprint y la velocidad generan el comportamiento de la variable de nivel BURDOWNCHARTS que se observa en la Figura 47. Las velocidades en cada uno de los sprints varían de acuerdo a la duración en horas y la cantidad de puntos asignados a dicho sprint.

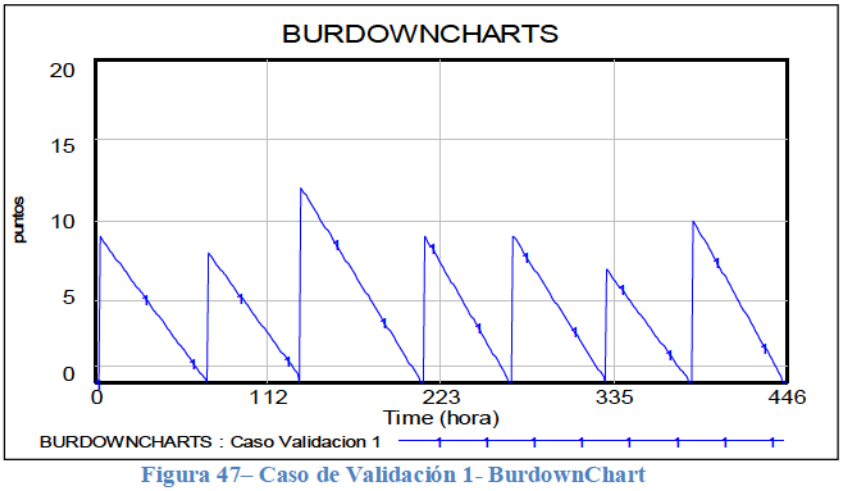


Figura 47– Caso de Validación 1- BurdownChart

En el párrafo anterior se mencionó que es necesaria una determinada velocidad de desarrollo de puntos de historia por día para poder completar las tareas planificadas, esta velocidad de trabajo diario se representa mediante la variable auxVelocidadIdeal.

En el presente caso de simulación los valores de la variable mencionada para cada sprint se representan en la Figura 48, donde aparecen superpuestos el inicio de cada sprint y la velocidad estimada para dicho Sprint.

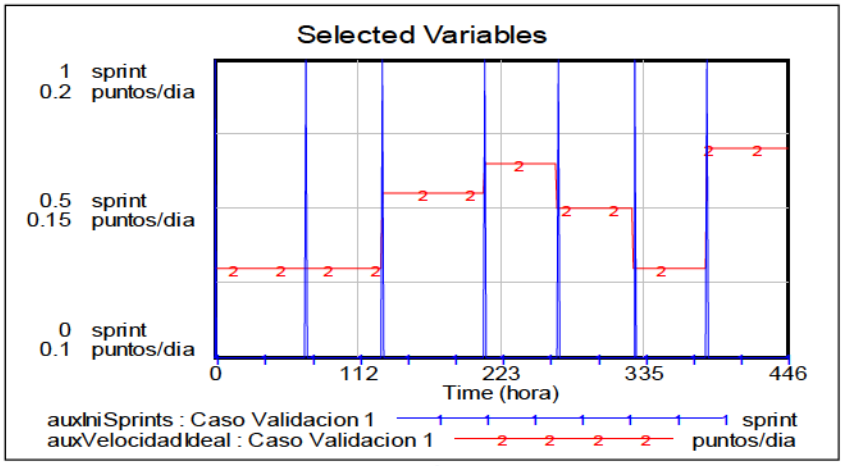


Figura 48 – Caso de Validación 1 - Velocidad Por Sprint

Como en el presente caso bajo simulación y según los valores de la Tabla 3 a cada punto de historia se le asignó una tarea, además no se expresan situaciones que retrasen el normal desarrollo de la codificación de las mismas, el avance de tareas y de puntos asignados se produce de manera gradual según la velocidad de desarrollo establecida.

El avance de las tareas y los puntos planificados pueden observarse en la Figura 49 donde de manera superpuesta están representadas las variables de nivel TAREASCODIFICADAS y PUNTOSPORSPRINT.

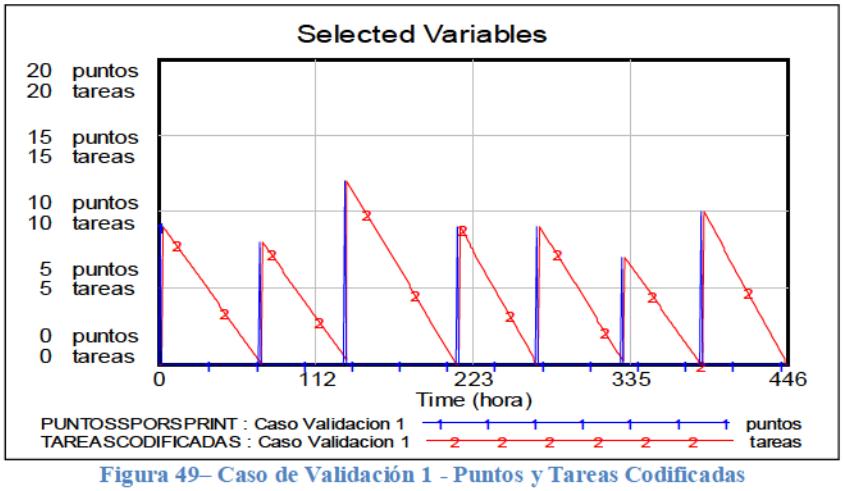


Figura 49– Caso de Validación 1 - Puntos y Tareas Codificadas

En la Figura 50 se presentan el Burdown Chart y las tareas que se fueron codificando durante el Sprint. Como se mencionó anteriormente al no existir retrasos por ausencias, o tareas a reprogramar, los valores de ambas figuras coinciden en su desarrollo, indicando que todo se desarrolló de acuerdo a lo planificado.

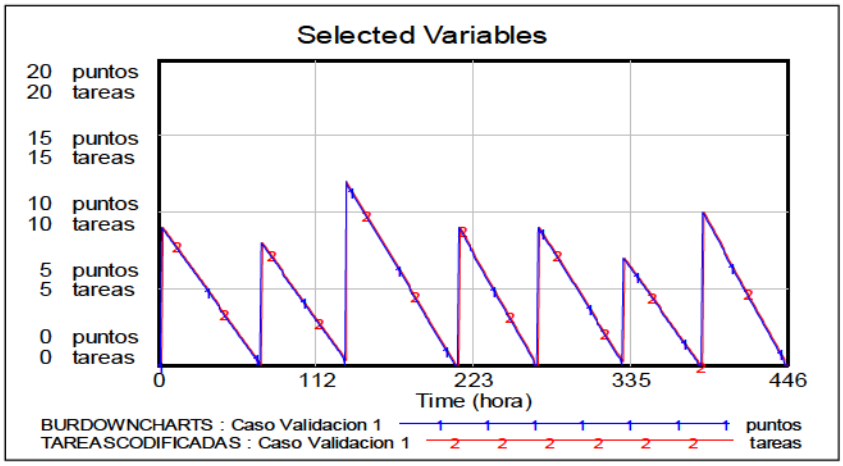


Figura 50 - Caso de Validación 1 –BurdownChart y Tareas Codificadas

En Figura 51 se presenta el número de pruebas de codificación a realizarse en cada sprint. Dado que por cada tarea a codificar se planifico una prueba, el comportamiento de la variable PRUEBASREALIZAR coincide con el número de tareas y puntos planificados para cada Sprint.

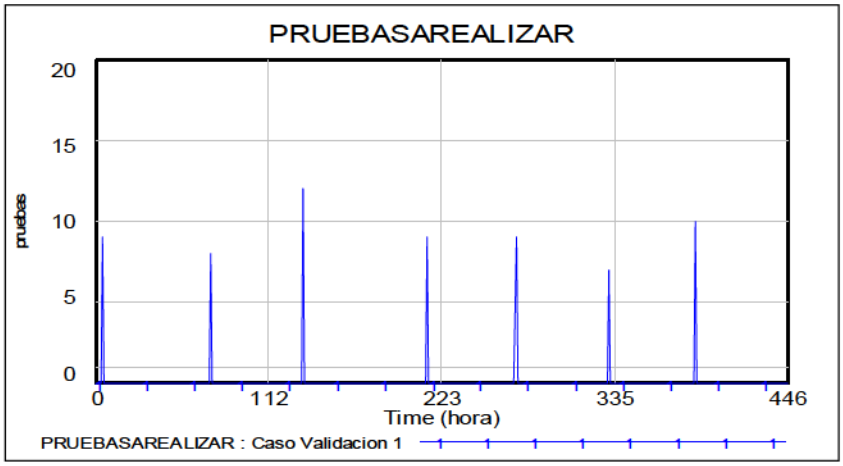


Figura 51– Caso de Validación 1 - Pruebas A Realizar Por Sprint

En la Figura 52 se observan de manera conjunta el resultado de las pruebas de codificación para aquellas tareas donde se detectaron errores y en aquellas que no, junto a las pruebas diseñadas.

Como para el presente caso no se especificaron situaciones de errores por cansancio o por presión, la totalidad de las tareas planificadas y codificadas no generaron pruebas con errores.

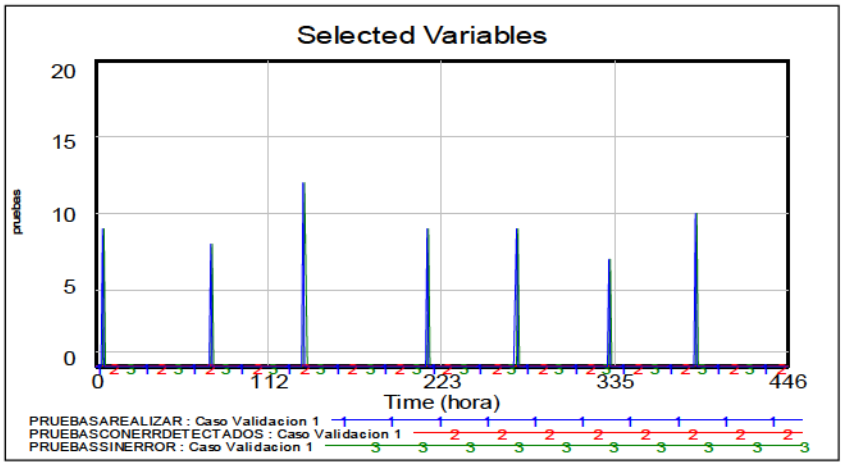


Figura 52 – Caso de Validación 1 - Pruebas Codificación Por Sprint

Al no estar expresados los errores de integración de funciones y tareas codificadas el número de tareas integradas fue del 100%. Este resultado puede apreciarse en la Figura 53 donde se muestran de manera conjunta y superpuesta las variables PRUEBASINTEGRACIONREALIZADAS, PRUEBASINTEGRADAS y PRUEBASCONERRINTEGRACION.

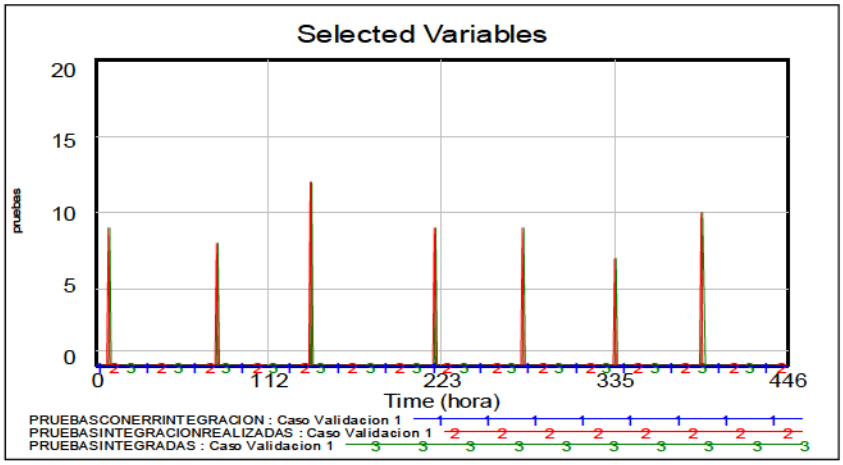


Figura 53 – Caso de Validación 1 - Pruebas de Integración Por Sprint

6.1.2 Caso 2: "Aplicação Do Processo Ágil de Gerenciamento Scrum No Desenvolvimento de Um Jogo Digital"

El trabajo desarrollado en Brasil tiene como finalidad la aplicación del método ágil Scrum al desarrollo de un juego digital. Para esto presenta un caso de estudio en el cual Scrum fue aplicado al proceso de creación de un juego de futbol 3D [43].

Además explica y presenta las principales características de Scrum, y presenta un esquema compuesto por una Contextualización, el estudio del caso y la conclusión.

El Proyecto tiene una meta de finalización en un plazo de 5(cinco) semanas. Dentro de las características del proyecto se deja en claro que no hace foco sobre la tecnología de desarrollo del juego, ni tampoco es objetivo hacer una comparación de diferentes metodologías de desarrollo.

Como datos relevantes, para la validación con este proyecto se respetó el tiempo máximo propuesto inicialmente en el trabajo de 930 horas hombre en tareas y de 200 horas hombre máximo por Sprint; se adicionaron 11 tareas extras en total, en los últimos 4 Sprints, lo que significó 110 puntos de historia más al proyecto. Estas tareas extras están incluidas entre las tareas planificadas.

Dado que el trabajo que se referencia sirve para validación en este apartado se presentan los requerimientos expresados en horas hombre de trabajo. Para la representación y validación de este caso se considera que 1(uno) punto de historia es equivalente a 10 (diez) horas hombre de trabajo

6.1.2.1 Datos Principales del Proyecto

En la Tabla 4 se pueden ver los valores de los parámetros utilizados en este caso de validación.

Tabla 4 – Caso de Validación 2 – Parámetros Iniciales

Sprint	Requerimientos	Puntos Historia	Puntos por Tareas Extras	Total Puntos	Duración en Horas	Velocidad Ideal Estimada	Inicio
1	10	19	0	19	40	0,475	0
2	12	18	2	21	40	0,525	40
3	14	19	4	23	40	0,579	80
4	14	24	2	26	40	0,659	120
5	11	23	3	26	72	0,378	160
Total	**61**	**103**	**11**	**114**	**232**	-	**232**

6.1.2.2 Otros Datos

- Cantidad de Integrantes: 6 Personas. 1 Scrum Master. 5 Team.
- Número de Sprints: 5(cinco).
- Duración Total del Proyecto: 232 horas.
- Requerimientos: 1 por cada punto.
- Pruebas por Requerimientos: 1 por cada Requerimiento.
- Horas Extras: No.
- Errores por Presión en el plazo: No.
- Errores de Integración: No.
- Promociones en el Team: No.
- Abandonos en el Team: No.
- Tareas Extras: Si.

6.1.2.3 Resultado de la Corrida

En función de los valores presentados en la Tabla 3, en la Figura 54 se presenta el inicio de cada uno de los Sprints planificados para este caso de validación.

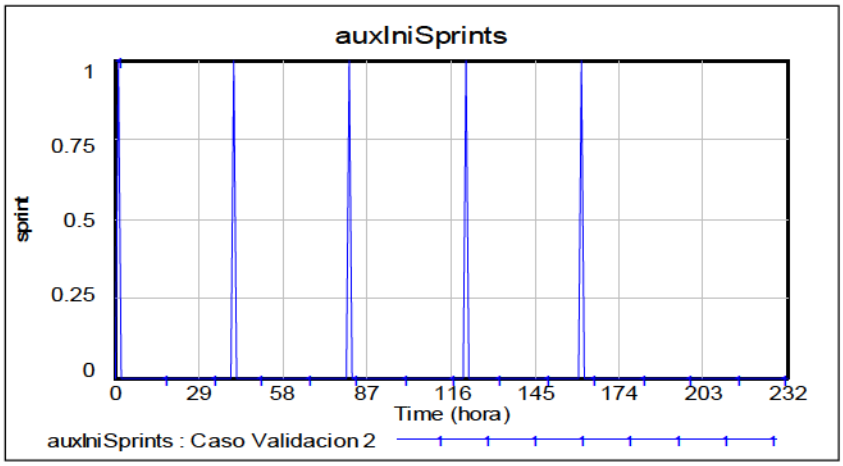


Figura 54 - Caso de Validación 2 - Inicio Sprint

La cantidad de puntos de Historia seleccionados para cada Sprint del presente escenario de validación, están establecidos en la variable lkpPuntosPorSprint y se reflejan en la variable PUNTOSPORSPRINT. Dichos valores se presentan en la Figura 55.

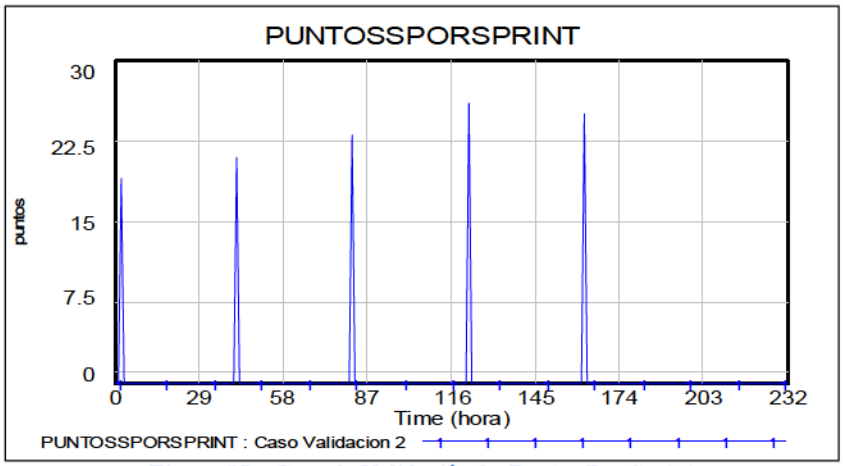


Figura 55 – Caso de Validación 2 - Puntos Por Sprint

De manera conjunta los puntos establecidos para cada Sprint y la velocidad de trabajo de desarrollo estimado generan la Figura 56.

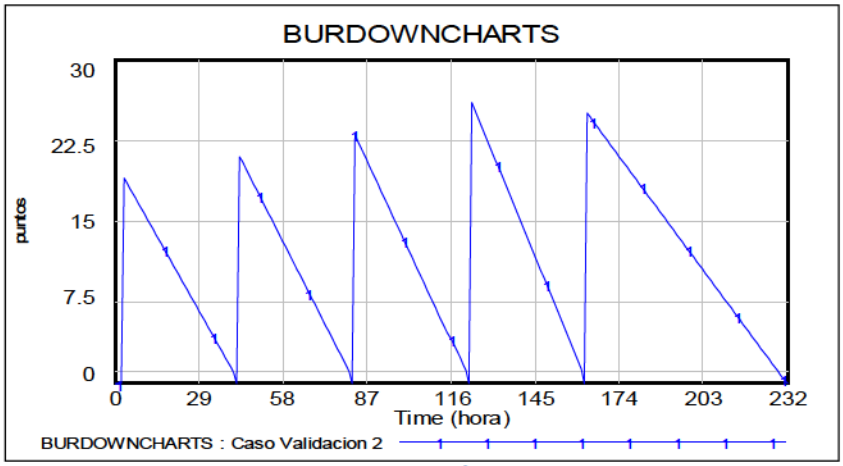


Figura 56 – Caso de Validación 2 – BurdownChart

Para poder arribar al final de cada Sprint con la totalidad de los puntos completados y que fueron planificados, el Team debe realizar diferentes tareas asociadas a cada historia de usuario.

La velocidad de cada Sprint se muestra en la Figura 57 de manera superpuesta con el Sprint correspondiente.

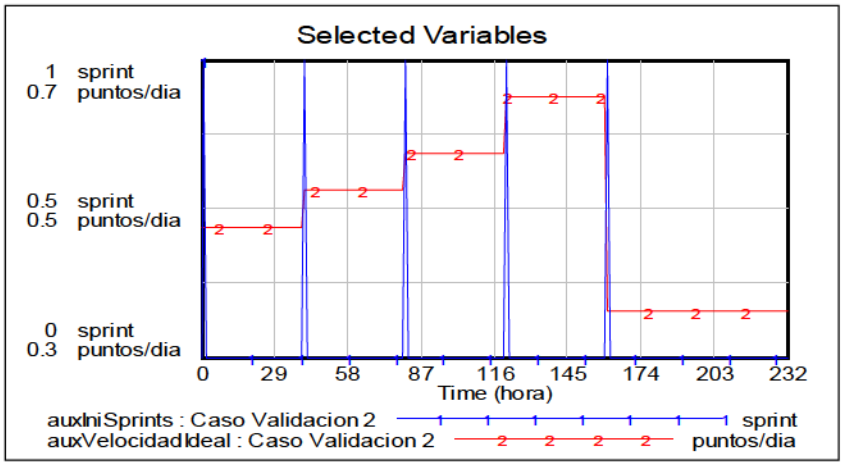


Figura 57 – Caso de Validación 2 - Velocidad Por Sprint

De igual manera al caso de validación anterior y según los valores de la Tabla 4 a cada punto de historia se le asignó una tarea, además no se expresan situaciones que retrasen el normal desarrollo de la codificación de las tareas, el avance de tareas y de puntos asignados se produce de manera gradual según la velocidad de desarrollo de tareas y en forma equitativa.

El avance de las tareas y los puntos completados pueden observarse en la Figura 58, donde de manera superpuesta están representados. Los valores de las tareas y de los puntos por hacer se almacenan en las variables de nivel TAREASCODIFICADAS y PUNTOSPORSPRINT.

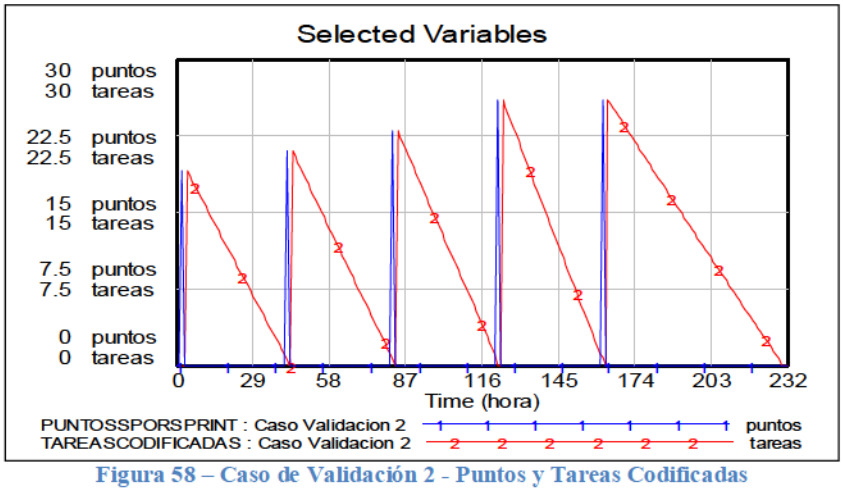


Figura 58 – Caso de Validación 2 - Puntos y Tareas Codificadas

En la Figura 59 se presentan el BurdownChart y las tareas que se fueron codificando durante el Sprint. Como se mencionó anteriormente al no existir retrasos por ausencias, o tareas a reprogramar los valores de ambos Figuras coinciden, indicando que se desarrolló todo de acuerdo a lo planificado.

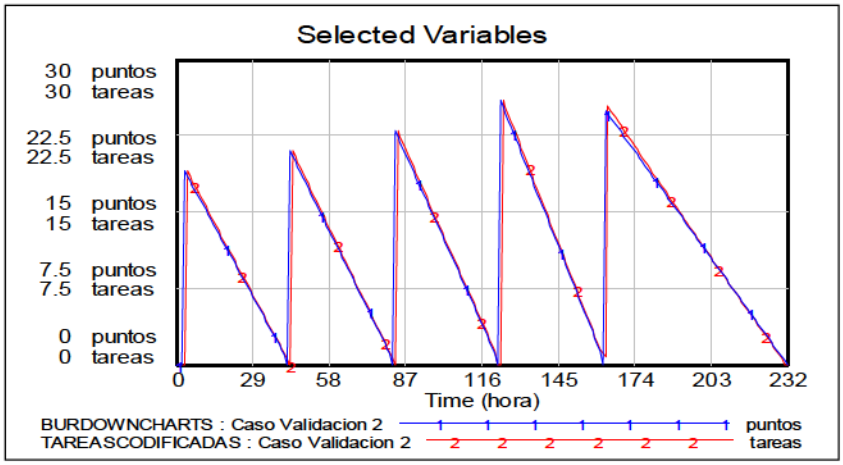


Figura 59 – Caso de Validación 2 - BurdownChart y Tareas Codificadas

En la Figura 60 se presentan las pruebas de codificación a realzarse en cada sprint.

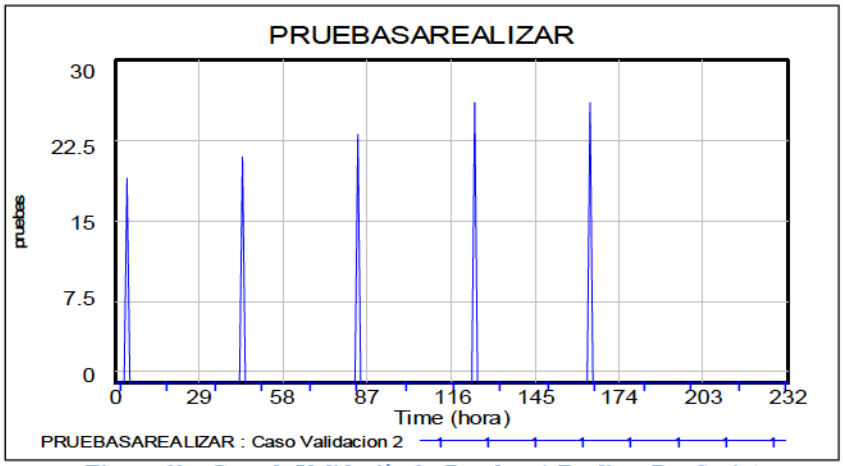


Figura 60 – Caso de Validación 2 - Pruebas A Realizar Por Sprint

En el presente caso no se especificaron situaciones de errores por cansancio o por presión. La totalidad de tareas codificadas no presentaron ningún tipo de error por lo tanto el nivel de tareas codificadas fue del 100%, esto puede apreciarse en la Figura 61

donde se muestran de manera conjunta y superpuestas las variables PRUEBASAREALIZAR, PRUEBASSINERROR y PRUEBASCONERRDETECTADOS.

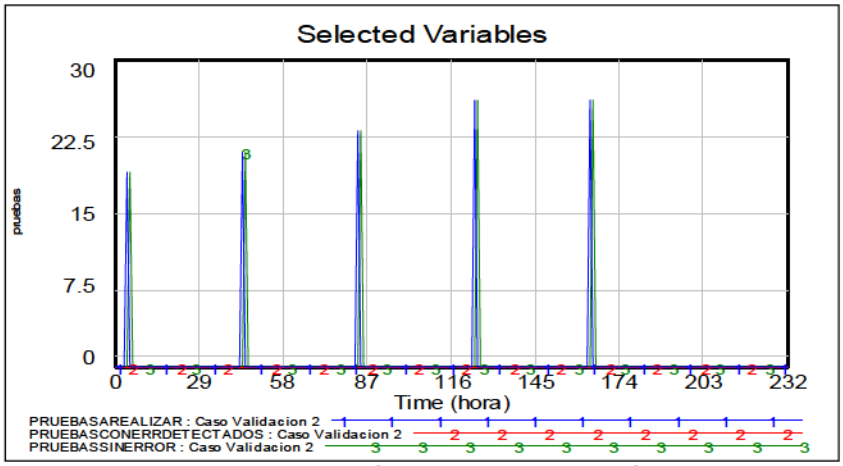


Figura 61 – Caso de Validación 2 - Pruebas Codificación Por Sprint

El nivel de tareas integradas fue del 100%, esto puede apreciarse en la Figura 62 donde se muestran de manera conjunta y superpuesta las variables PRUEBASINTEGRACIONREALIZADAS, PRUEBASINTEGRADAS y PRUEBASCONERRINTEGRACION.

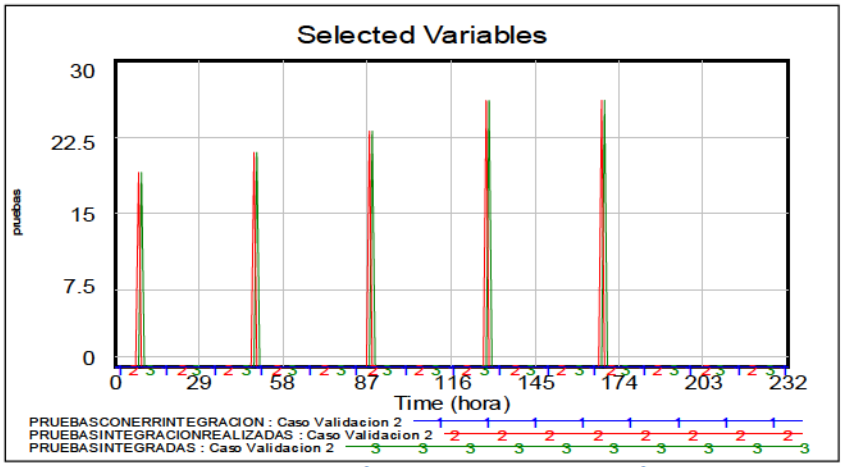


Figura 62 - Caso de Validación 2 - Pruebas de Integración Por Sprint

6.1.3 Caso 3: "Método Ágil Scrum aplicado al desarrollo de un software de trazabilidad"

El trabajo consiste en aplicar el método ágil Scrum al desarrollo de un Software de Trazabilidad. El presente proyecto surge de la solicitud realizada por un productor de uva quien necesitaba de un Software de Trazabilidad que le facilite la tarea de registración de todas las operaciones realizadas durante el proceso de producción de cada una de sus propiedades, y que le permita optimizar el manejo de información en el campo y demostrar que se cumple con la trazabilidad [44].

Básicamente el software desarrollado debe mantener un registro de las propiedades rurales, identificando para cada una de ellas sus titulares y/o productores, y para cada viñedo la distribución de sus cuarteles, origen de los mismos, el tipo de conducción y las variedades de uva que contempla. Además debe administrar datos de titulares y/o productores, trabajadores, proveedores y bodegas. Administrar las maquinarias y camiones, con sus revisiones técnicas y seguros de transportes correspondientes.

Dentro de lo que se considera relevante en este trabajo, el caso de validación presenta la característica de que tiene 2 integrantes, donde 1 desarrolla el rol de Scrum Master y forma parte del Team, el otro miembro del equipo efectuará los roles de Product Owner, Cliente y Usuario final del sistema. El tiempo que se dedicará al mismo es una jornada de medio día, 20 horas semanales aproximadamente.

Otro dato relevante es que el proyecto en sus inicios fue estimado en 464 horas y consumió 488 horas, por lo tanto, la entrega al cliente del último Sprint se hizo una semana después de lo presupuestado al comienzo del proyecto.

6.1.3.1 Datos Principales del Proyecto

En la Tabla 5 se expresan los valores utilizados para la simulación realizada para la validación de este caso.

Tabla 5 – Caso de Validación 3 – Parámetros Iniciales

Sprint	Requerimientos	Puntos Historia	Puntos por Tareas Extras	Total Puntos	Duración en Horas	Velocidad Ideal Estimada	Inicio
1	2	64	0	64	64	1,000	0
2	10	80	0	80	80	1,000	64
3	9	80	0	80	80	1,000	144
4	10	82	0	82	82	1,000	224
5	9	80	0	80	80	1,000	306
6	11	102	0	102	102	1,000	388
Total	**51**	**488**	**0**	**488**	**488**	**-**	**488**

6.1.3.2 Otros Datos

- Integrantes: 2 Personas. 1 Scrum Master. 2 Team.
- Experiencia: No especifica.
- Número de Sprints: 6.
- Duración de Cada Sprint: Variable
- Duración Total del Proyecto: 488horas.
- Velocidad Estimada de Cada Sprint: Variable
- Horas Diarias: 4 horas.
- Horas Hombre Ideal : 1 hora por 1 punto historia
- Requerimientos: 1 por cada punto de historia.
- Pruebas por Requerimientos: 1 por cada Requerimiento.
- Horas Extras: No
- Errores por Presión en el plazo: No.
- Errores de Integración: No.
- Promociones en el Team: No.
- Abandonos en el Team: No.
- Tareas Extras: No.

6.1.3.3 Resultado de la Corrida

En las Figuras que completan el presente caso de validación pueden verse entre otras las Figuras de requerimientos por Sprint, BurdownChart, Tareas Extras y la comparativa entre los requerimientos a completar y las tareas completadas respectivamente.

La Figura 63 presenta el momento de inicio de cada Sprint a lo largo del proyecto.

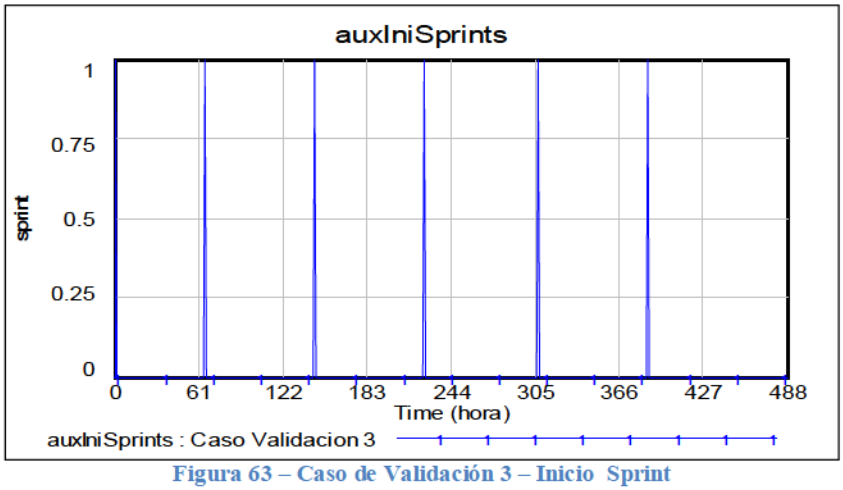


Figura 63 – Caso de Validación 3 – Inicio Sprint

La cantidad de puntos de Historia seleccionados para cada Sprint establecidos en la variable lkpPuntosPorSprint y que se reflejan en la variable auxPuntosPorSprint se presentan en la siguiente Figura 64.

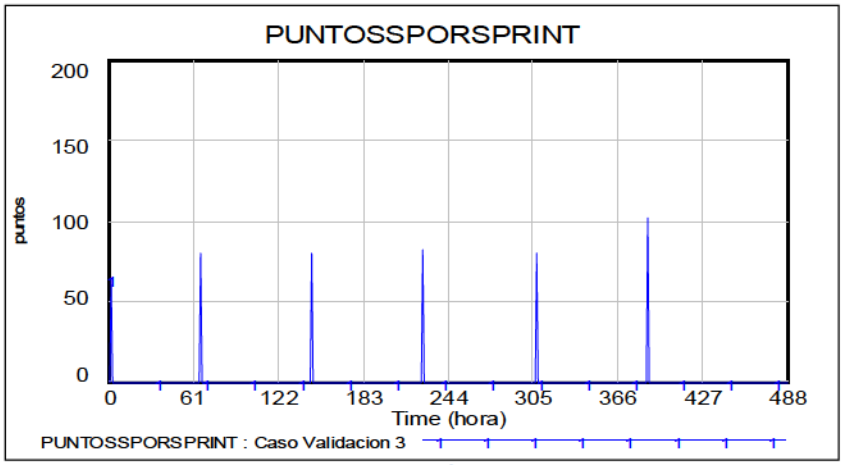


Figura 64 – Caso de Validación 3 – Puntos Por Sprint

Como se mencionó anteriormente la variable auxPuntosPorSprint indica la cantidad de puntos a desarrollar en cada uno de los sprints. Para poder arribar al final de cada Sprint con la totalidad de los puntos completados el Team debe realizar diferentes tareas asociadas a cada historia de usuario, la velocidad con la que estas tareas se realizan permiten o no, terminar el Sprint con la totalidad de los puntos realizados.

De manera conjunta, los puntos establecidos al inicio del sprint y la velocidad de trabajo establecida inicialmente generan la Figura 65. Las velocidades en cada uno de los sprints varían de acuerdo a la duración en horas y la cantidad de puntos asignados a dicho sprint.

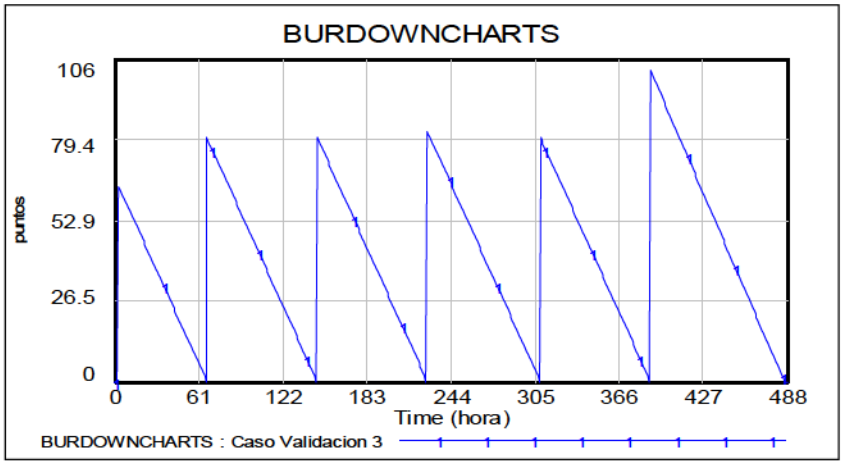


Figura 65 – Caso de Validación 3 – BurdownChart

Como se mencionó en párrafos anteriores es necesaria cierta velocidad de desarrollo de puntos de historia por día, y para representar esta velocidad se utiliza la variable auxVelocidadIdeal. En el presente caso bajo simulación la velocidad para cada uno de los sprints se representa en la Figura 66.

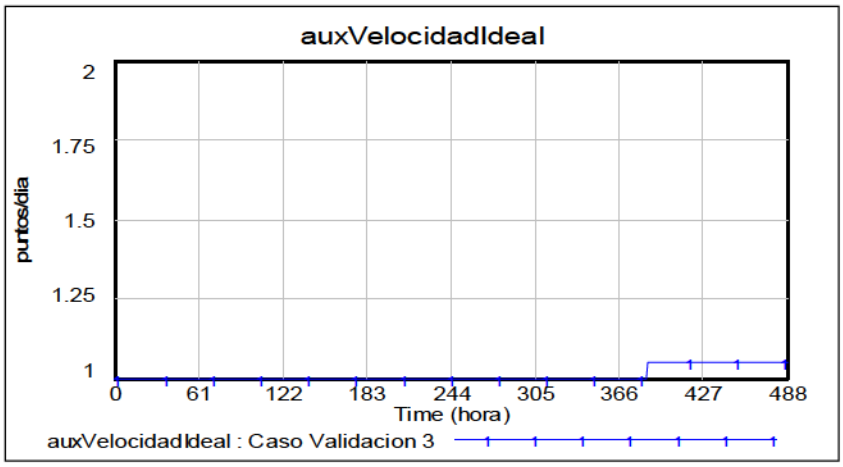


Figura 66 – Caso de Validación 3 – Velocidad Por Sprint

Como en el presente caso bajo simulación y según los valores de la Tabla 3 a cada punto de historia se le asignó una tarea, además no se presentan situaciones que retrasen el normal desarrollo de la codificación de las tareas, el avance de tareas y de puntos asignados se produce de manera normal y según la velocidad de desarrollo de tareas estimadas al inicio.

El avance de las tareas y los puntos planificados pueden observarse en la Figura 67, donde de manera superpuesta están representados.

Los valores de las tareas y de los puntos por hacer se almacenan en las variables de nivel TAREASCODIFICADAS y PUNTOSPORSPRINT respectivamente.

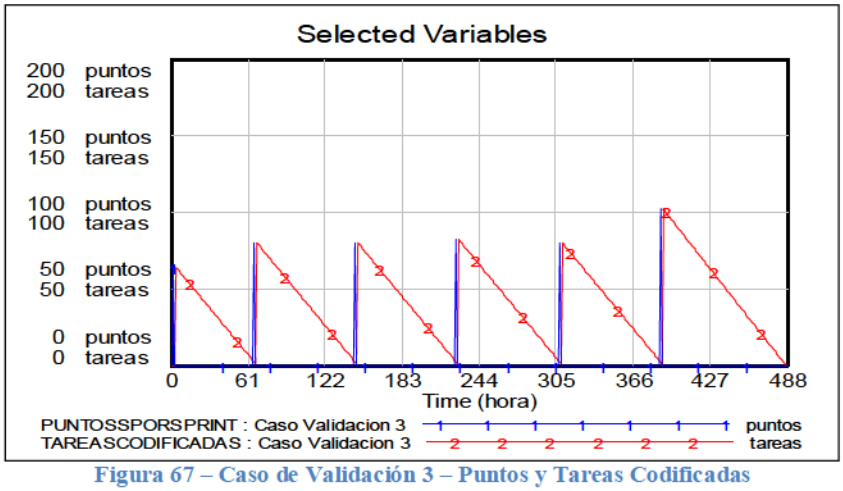


Figura 67 – Caso de Validación 3 – Puntos y Tareas Codificadas

En la Figura 68 se presentan el BurdownChart y las tareas que se fueron codificando durante el Sprint. Como se mencionó anteriormente al no existir retrasos por ausencias, o tareas a reprogramar, los valores de ambas Figuras coinciden, indicando que se desarrolló todo de acuerdo a lo planificado.

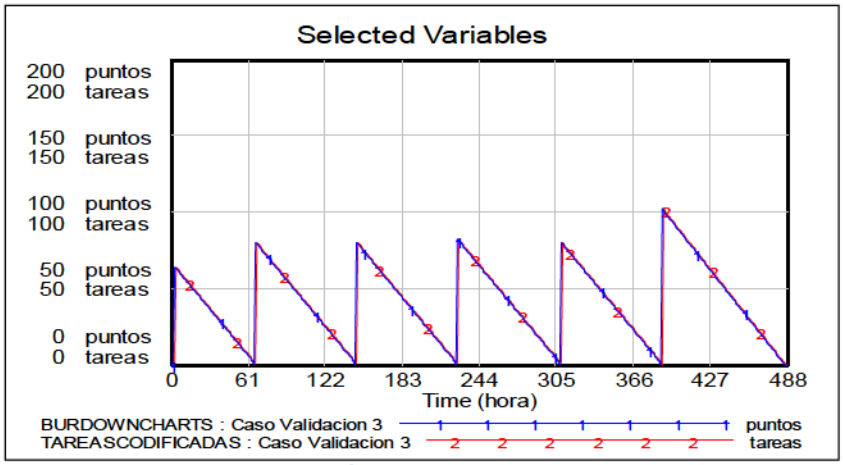


Figura 68 - Caso de Validación 3 – BurdownChart y Tareas Codificadas

En la Figura 69 se presentan las pruebas de codificación a realizarse en cada sprint.

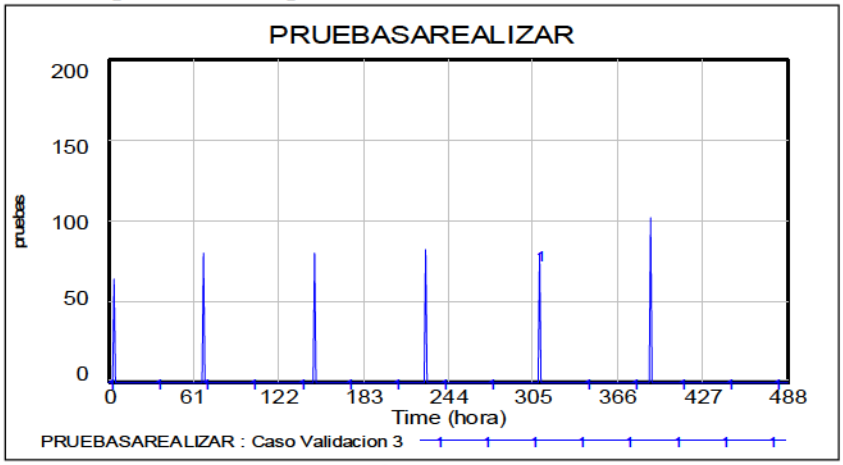


Figura 69 - Caso de Validación 3 – Pruebas A Realizar Por Sprint

En el presente caso de validación no se especificaron situaciones donde se pudieran presentar errores por cansancio o por presión, la totalidad de tareas codificadas fue del

100%. El resultado de la corrida para esta situación, puede apreciarse en la Figura 70 donde se muestran de manera conjunta y superpuesta los valores asumidos por las variables PRUEBASAREALIZAR, PRUEBASSINERROR y PRUEBASCONERRDETECTADOS.

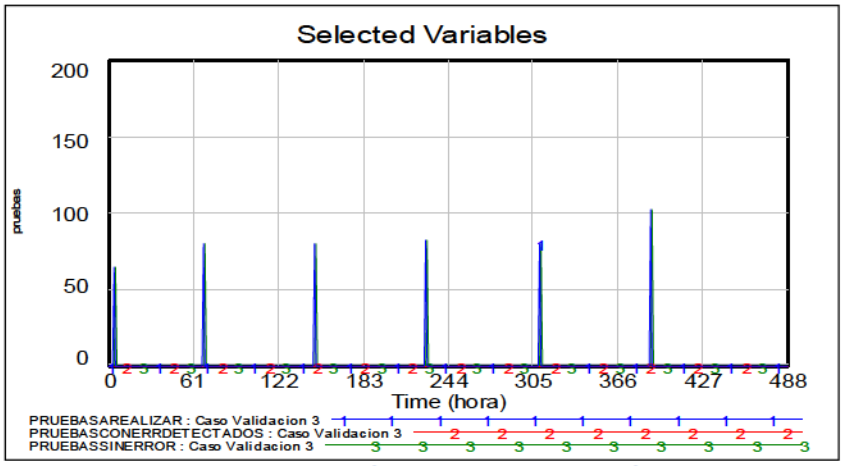


Figura 70 - Caso de Validación 3 – Pruebas Codificación Por Sprint

El nivel de tareas integradas fue del 100%, esto puede apreciarse en la Figura 71 donde se muestran de manera conjunta y superpuestas las variables PRUEBASINTEGRACIONREALIZADAS, PRUEBASINTEGRADAS y PRUEBASCONERRINTEGRACION, donde al no existir errores de integración la última variable no presenta ningún valor y se mantiene en cero. Por el contrario las primeras dos variables presentan exactamente el mismo comportamiento.

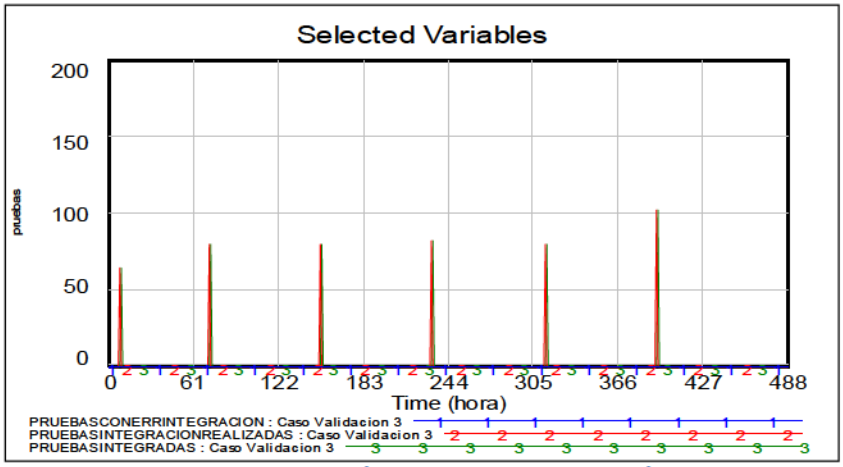


Figura 71 – Caso de Validación 3 – Pruebas de Integración Por Sprint

6.2 Casos de Experimentación realizados

En esta sección se describen los casos de experimentación realizados con el modelo. Si bien se tratan de propuestas de casos artificiales, los mismos representan situaciones típicas en proyectos de Scrum. Se presentaran cuatro experimentos los cuales van en complejidad creciente, en cuanto la cantidad de variables y subsistemas intervinientes. En cada uno de estos casos se presentará una situación base o ideal, la situación modificada real luego de la corrida y dos propuestas de modificación de políticas con el fin ver qué pasaría si se adoptaran las mismas.

6.2.1 Parámetros Comunes para todos los experimentos

Para la realización de los diferentes casos experimentales existen en el modelo diferentes variables tipo LookUp que se utilizan para determinar el comportamiento de otras variables y el modelo.

A continuación en la Tabla 6 se presentan los valores que se asignaron a los coeficientes y se utilizaron en las variables LookUp. Es válido destacar que los valores establecidos en las variables lkpCoeficienteCansancio, lkpCoeficienteCansancioDiario y lkpTasaPresionPlazo se adaptaron de [45] y [46], aunque podrán ser modificados o establecidos por el Scrum Master.

Tabla 6 – Parámetros Comunes para todos los experimentos

lkpCoeficienteCansancio		lkpPresionPlazo		lkpCoeficienteCansancioDiario	
x...	y...	x...	y...	x...	y...
0	0	**0**	0,10	**1**	0,01
40	0,10	**0,1**	0,10	**2**	0,01
80	0,15	**0,2**	0,10	**3**	0,02
120	0,20	**0,3**	0,15	**4**	0,03
160	0,25	**0,4**	0,20	**5**	0,04
200	0,30	**0,5**	0,25	**6**	0,05
240	0,35	**0,6**	0,30	**7**	0,06
280	0,40	**0,7**	0,40	**8**	0,07
320	0,45	**0,8**	0,50	**9**	0,08
360	0,50	**0,9**	0,50	**10**	0,09
400	0,52	**1**	0,50	**11**	0,10
440	0,55				
480	0,57				
520	0,62				
600	0,70				

En función de los valores de la Tabla 6 las diferentes variables tomaron comportamientos diferentes. A continuación se presentan los gráficos de dichas variables Lookups, Cansancio, Presión de Plazo y Cansancio Diario en la Figura 72, Figura 73 y Figura 74 respectivamente.

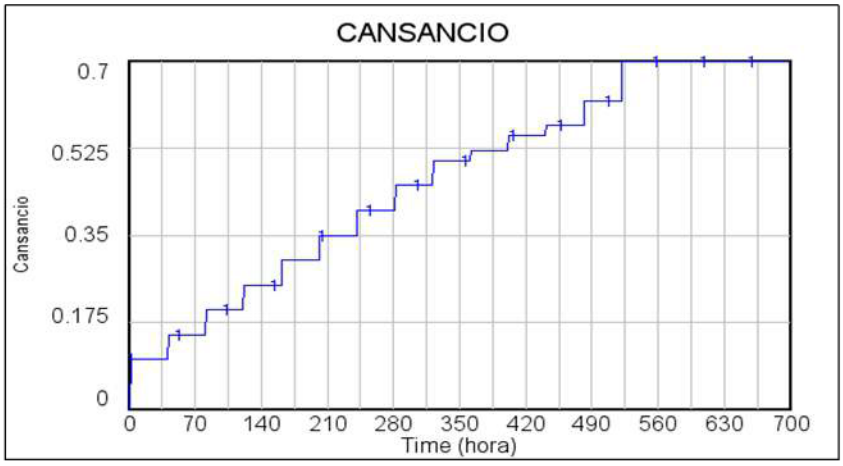


Figura 72 - Comportamiento Coeficiente Cansancio

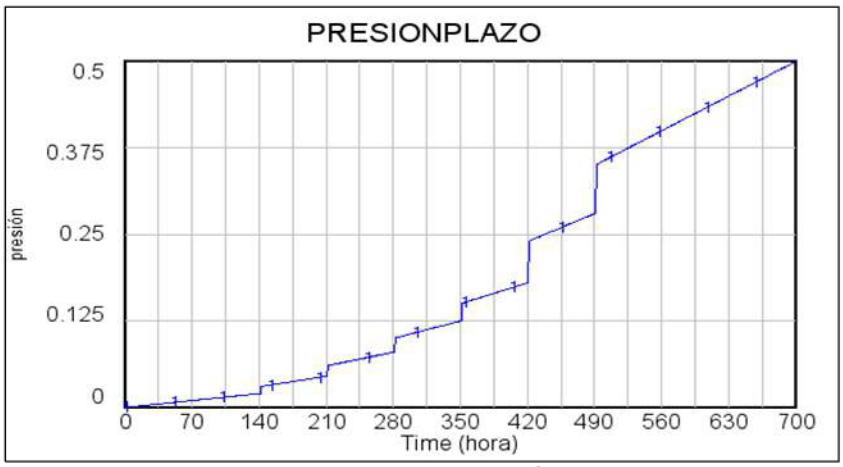


Figura 73 – Comportamiento Presión en el Plazo

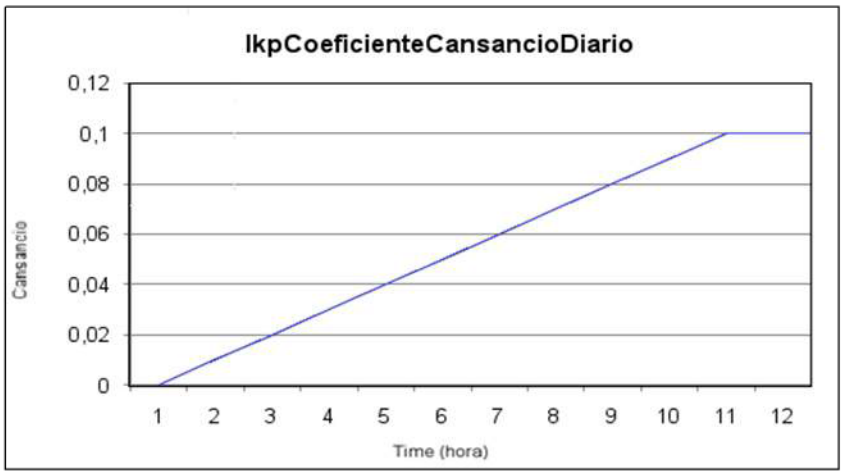


Figura 74 – Comportamiento Coeficiente Cansancio Diario

Para la generación de los valores aleatorios de las variables del modelo donde se utiliza la función Random, se seleccionaron las siguientes semillas:

auxIntegrantesAusentes: 555555.
auxNroCandidatos: 12301.

Además para los experimentos que así lo requieran se detallan a continuación los valores que toman los parámetros del Subsistema de Adquisición de Experiencia de Recursos Humanos.

tasaAprendizajeSeniors: 0.8.
tasaAprendizajePromocionales: 0.4.
tasaAprendizajeJrsContratados: 0.1.
auxNivelProductividadSeniors: 0.9.
auxNivelProductividadJuniorsPromos: 0.4.
auxNivelProductividadJuniors: 0.1.

6.2.2 Caso de Experimentación 1

Este primer caso experimental es un proyecto de mediana complejidad en donde se consideran la aparición de tareas extras y de errores por presión en el plazo a lo largo del proyecto que comprometerían la finalización del mismo.

6.2.2.1 Parámetros Generales

La Tabla 7 presenta los valores de los parámetros generales, establecidos por el scrum master y el Team, que permiten calcular los valores de la Tabla 8, y parámetros por sprint para utilizados en este experimento. Esta es la situación base.

Tabla 7 – Experimento 1 – Parámetros Generales

Variable	Valor
Horas Diarias	8
Días semana	5
Horas.Extra por Semana	1
Horas Totales por Semana	41
Horas Totales Normales	200
Horas Totales Extras	5
Horas Totales Proyecto	205
Sprints	5
Puntos de Historia	195
Pruebas Por Tarea	1
Velocidad Prueba	1
Inasistencias Pactadas	15
Inasistencias Imprevistas	0
Factor Dedicación Sprint	60,0%
Factor Dedicación Diario	80,0%
Días Hombre Ideal	60
Días Hombre Real	51
Team	4
Errores por Presión en el plazo	Si
Errores de Integración	No
Promociones en el Team	No
Abandonos en el Team	No
Tareas Extras	Si

6.2.2.2 Parámetros Iniciales Experimento

La Tabla 8 presenta los valores iniciales utilizados para la corrida de este escenario.

Tabla 8 – Experimento 1 – Parámetros por Srpint

Sprint	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
1	40	40	0,976	0	1	41	0	40	0	0	0
2	38	40	0,927	40	1	41	3	41	0	0	44
3	40	40	0,976	80	1	41	0	40	0	0	87
4	39	40	0,951	120	1	41	3	42	0	0	130
5	38	40	1.030	160	1	41	3	41	0	0	175
Total	**195**	**200**			**5**	**205**	**9**	**204**	**0**	**0**	

El detalle de cada columna es el siguiente:

a) Puntos Historia: Determinado por el Team.
b) Duración planificada en horas: Surge de multiplicar las horas normales de trabajo por la cantidad de días semanales de trabajo.
c) Velocidad Estimada: Inicialmente surge de la división de (a) sobre (b). Luego se realizan ajustes para generar el BurdownChart
d) Inicio Planificado: Suma acumulada de los diferentes valores de (b).
e) horas extras: Determinadas según la necesidad del Team. Expresa las horas extras semanales.
f) Duración con Horas. Extras: Surge de la suma de (b) más (e).
g) Tareas Extras: Determinadas por el Scrum Master en función de la necesidad del Team.
h) Puntos con Tareas Extras: Surge de sumar (a) más (g).
i) Tareas Extras No Planificadas: Generado Aleatoriamente.
j) Ausencias No acordadas: Generado Aleatoriamente.
k) Inicio Recalculado: Suma acumulada de los diferentes valores de (f).

6.2.2.3 Resultado del experimento

La Figura 75 presenta el momento en que se inician los diferentes Sprints a lo largo del proyecto.

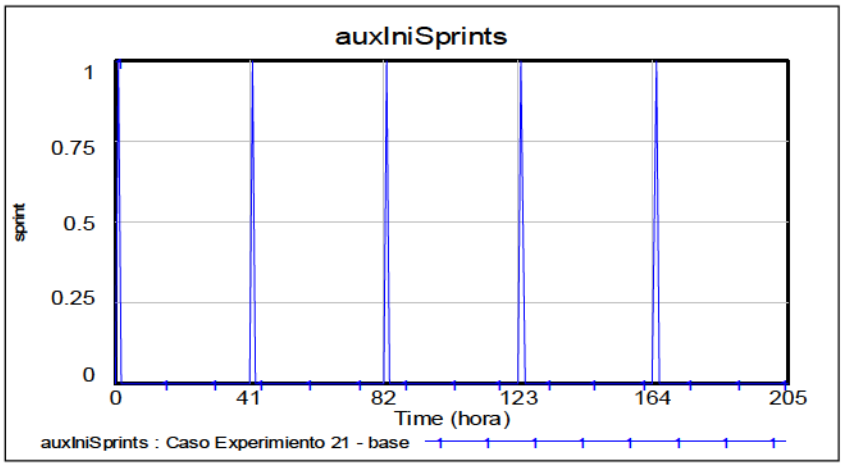


Figura 75 - Experimento 1 – Inicio Sprints

Según lo planificado para el experimento y los valores de la tercera columna de la Tabla 7 se generó la siguiente Figura 76 donde se presentan los puntos planificados por Sprint.

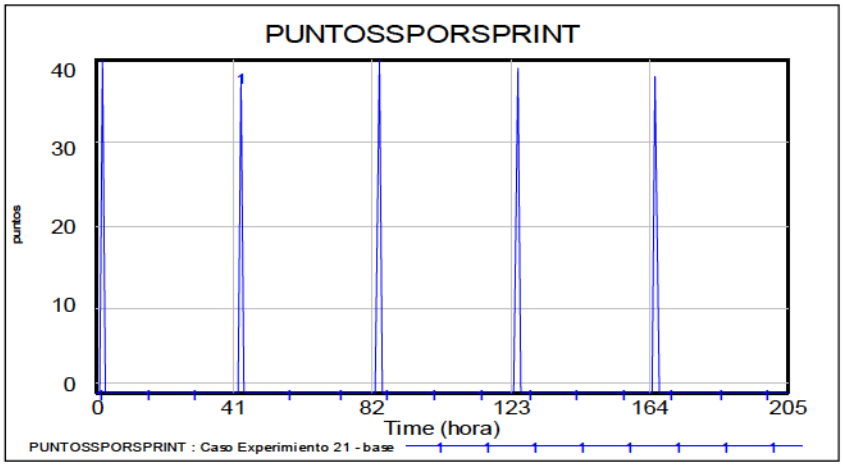


Figura 76 – Experimento 1 – Puntos Por Sprint

En la Figura 77 en función de los puntos planificados y de la velocidad estimada, se generan los valores de la variable BurdownCharts, que muestra el avance ideal del proyecto a partir de completar los puntos planificados según la velocidad ideal estimada.

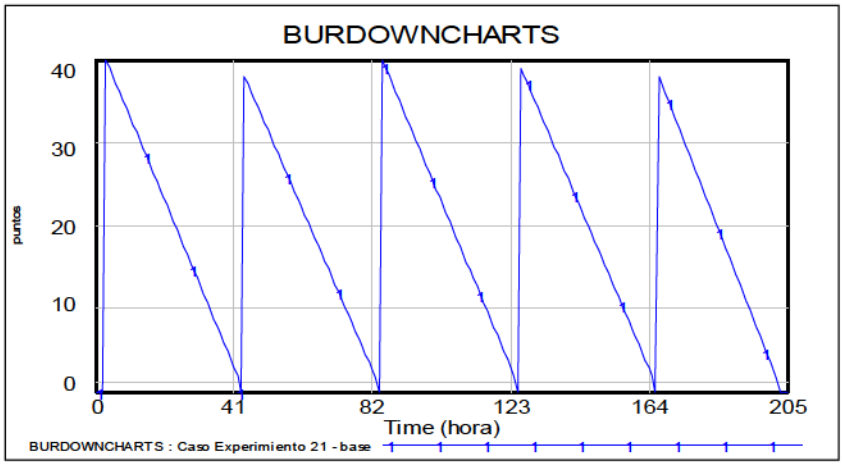


Figura 77 - Experimento 1 – Burdownchart

En la Figura 78 se muestra la velocidad inicial estimada para los Sprints del caso experimental, según lo determinado y establecido en la cuarta columna de la Tabla 8.

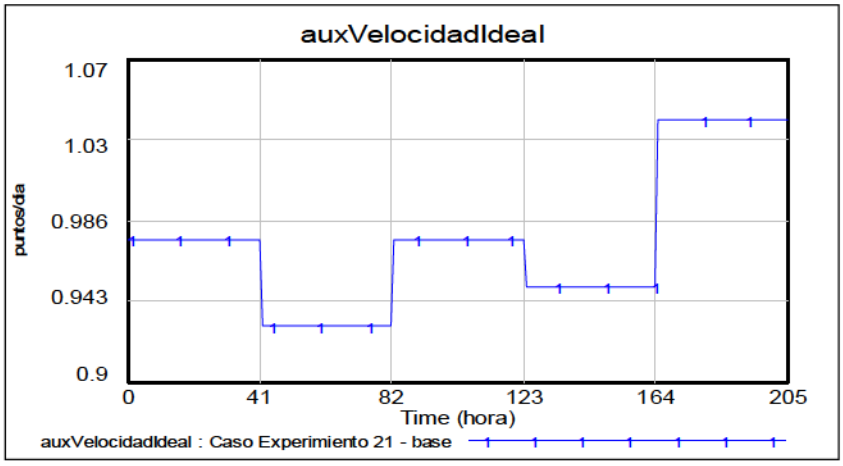


Figura 78 – Experimento 1 – Velocidad Ideal Planificada

Según los puntos planificados para el experimento y representados en la variable PUNTOSPORHACER se generó la Figura 79 donde se presentan los puntos que deberían ir completándose durante cada Sprint.

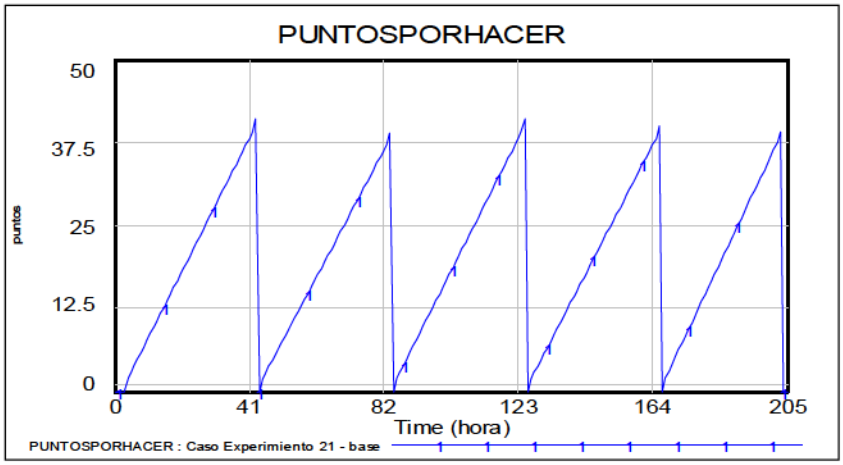


Figura 79 – Experimento 1 – Puntos Por Hacer

Es este caso a cada punto de historia le corresponde una tarea, es por ello que en la Figura 80 se observa que el comportamiento de la variable TAREASPENDIENTESCODIF se corresponde con el de la variable PUNTOS POR SPRINT.

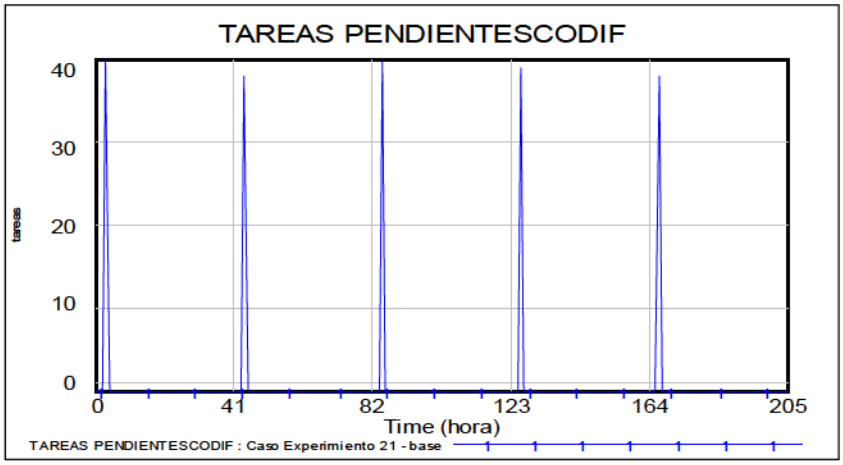


Figura 80 - Experimento 1 – Tareas Pendientes de Codificación

En la Figura 81 y la Figura 82 se observan los momentos a lo largo del proyecto en los se presentan diferentes tareas a recodificar, ya sean aquellas tareas extras a desarrollar planificadas inicialmente o aquellas que hayan presentado errores de codificación.

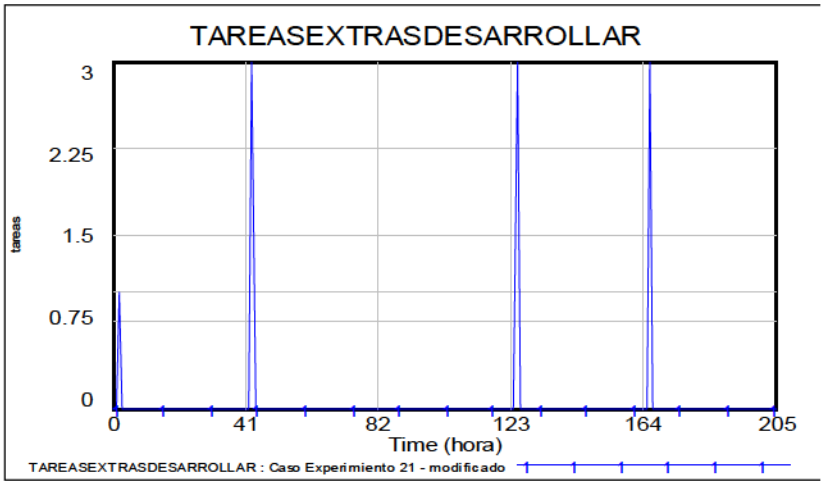


Figura 81 – Experimento 1 – Tareas Extras

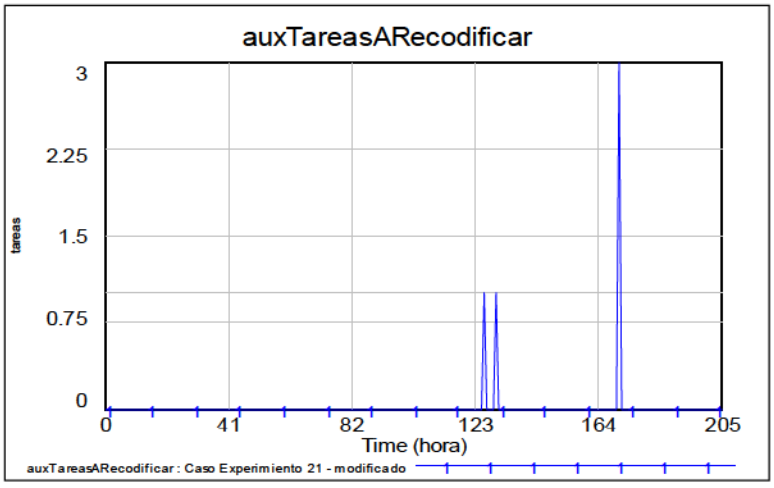


Figura 82 – Experimento 1 – Tareas A Recodificar

A raíz de las tareas con errores y las tareas extras planificadas que surgieron a lo largo del proyecto, la cantidad de tareas pendientes de codificación estimadas inicialmente aumentan con respecto a la situación base. En la Figura 83 se muestran las tareas pendientes de codificación iniciales (vistas en la Figura 80) y de manera superpuesta el resultado de sumarle a esas tareas pendientes las tareas extras a desarrollar.

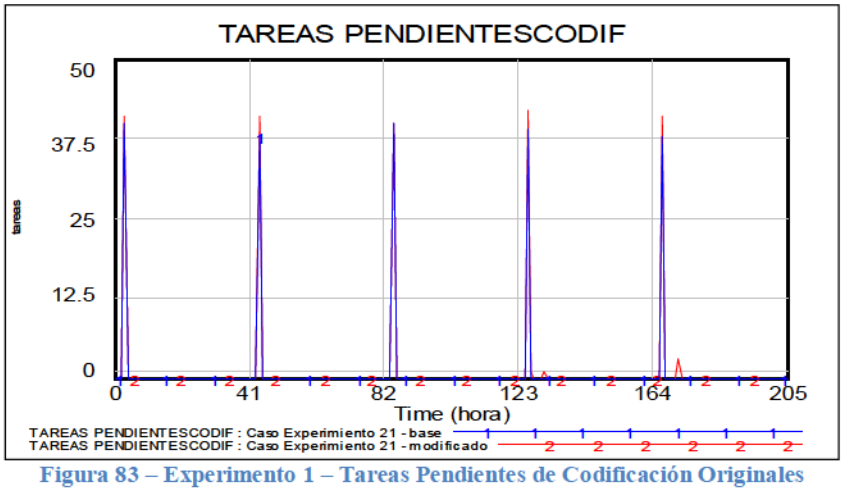


Figura 83 – Experimento 1 – Tareas Pendientes de Codificación Originales y con Tareas Extras

En la Figura 84 se observa el avance ideal de cómo se desarrollaría la codificación de las tareas planificadas durante cada uno de los Sprints. En este caso se incluyen las tareas planificadas en la situación base y las tareas a las que se sumaron las tareas adicionales en la situación modificada real.

Una característica que presenta la Figura 84 es que en la referencia 2 se observa que la línea que muestra el avance de la codificación durante los Sprints no finaliza en el punto cero, y no se completará el proyecto con los tiempos establecidos en la situación base, lo que indica que aún quedan tareas pendientes por codificar. Esto se debe a que en dicha corrida se adicionaron tareas y se mantuvo la velocidad establecida inicialmente en Tabla 8.

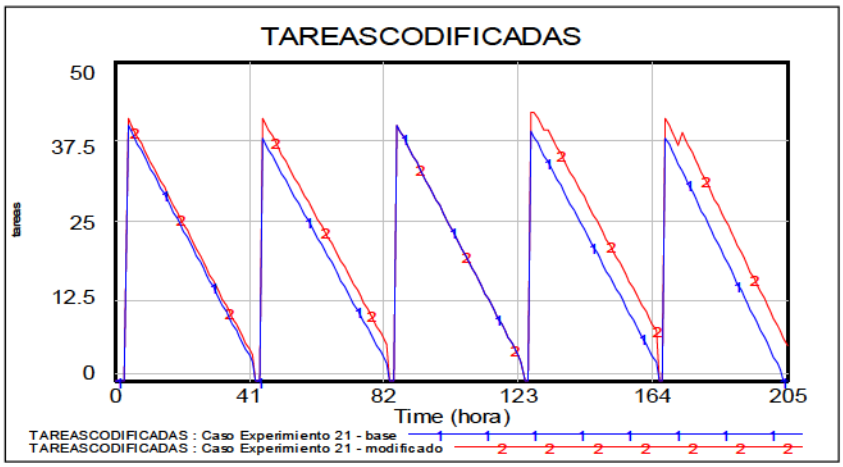


Figura 84 – Experimento 1 – Tareas Codificadas

En la Figura 85 se presentan las diferentes pruebas a diseñar en función de las tareas codificadas. Se incluyen en la figura las tareas planificadas en la situación base y la situación modificada real que contempla las tareas adicionales. Como ocurrió al momento de la codificación, aquí también se aprecia que la totalidad de lo planificado (referencia 2) no llega a completarse al momento de la finalización de los Sprints dado que se mantuvo la velocidad inicial estimada.

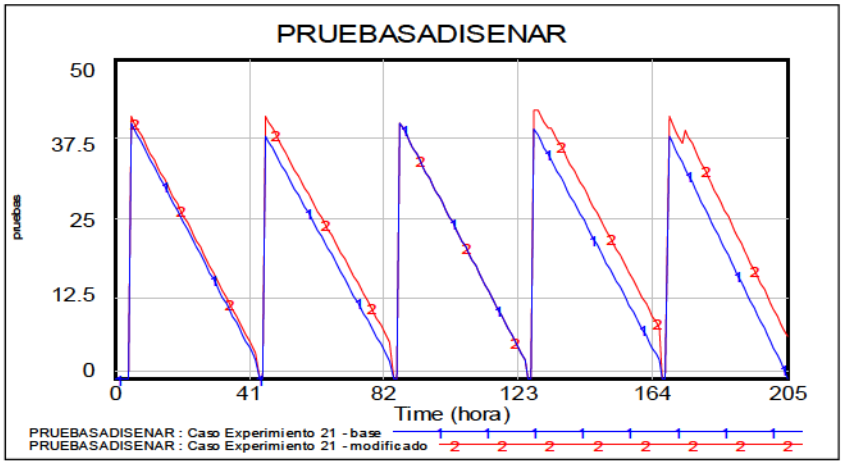


Figura 85 – Experimento 1 – Pruebas A Diseñar

Como ya se mencionó en el presente Experimento se tienen en consideración aquellas situaciones donde existen tareas con errores de codificación. En la Figura 86 se presentan de manera superpuesta el resultado de las pruebas Sin errores de codificacion y las Tareas que presentaron errores de codificacion. Se observa en la figura que en los momentos donde se detectaron los errores el normal descenso de la gráfica que corresponde al desarrollo de las tareas presenta bajas o caídas, resultantes de restar estas tareas con errores a las tareas totales desarrolladas.

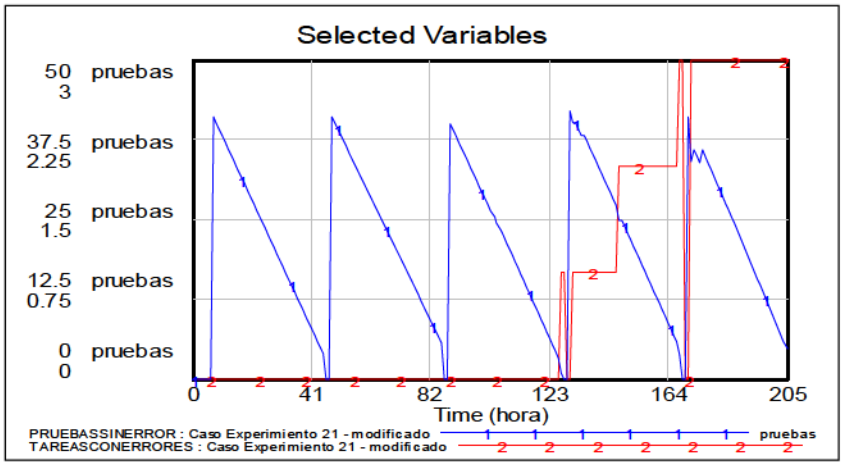


Figura 86 – Experimento 1 – Pruebas y Tareas Con Errores

En la Figura 87 se muestra el comportamiento de la variable que representa las pruebas de integración realizadas. En dicha figura se observa el comportamiento del modelo con las tareas planificadas inicialmente, y el comportamiento que tuvo al momento de considerar las tareas extras planificadas y de restar aquellas tareas con errores al total planificado.

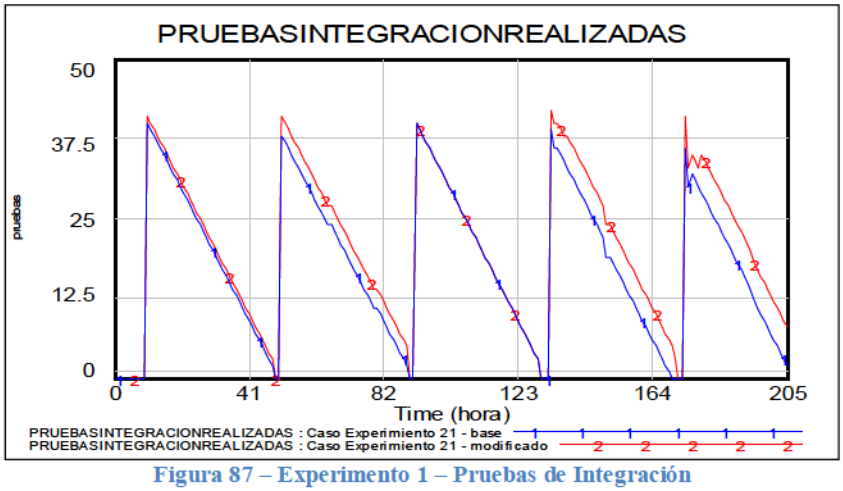


Figura 87 – Experimento 1 – Pruebas de Integración

6.2.2.4 Políticas Propuestas

Dada la situación inicial y presentada en la Tabla 8 y la situación modificada real luego de la corrida, se observa que en algunos Sprints la cantidad de horas establecidas para estos, exceden a las horas planteadas en la situación base ideal.

Para resolver la situación planteada se presentan dos políticas alternativas. La primera de ellas, propone mantener las duraciones originales de los Sprints de la situación base y realizar un ajuste porcentual a la velocidad de desarrollo que se estableciera originalmente. El resultado de la aplicación de esta política se puede ver en la Figura 88, Figura 90, Figura 92 y Figura 94 que representan las variables del modelo afectadas.

La segunda política propone, establecer nuevas duraciones de los Sprints y mantener la velocidad estimada para la situación base, esto está representado en la Figura 89, Figura 91, Figura 93 y Figura 95, donde se destaca la mayor duración del proyecto que se extendió a 205 horas normales.

A continuación se presentan aquellas variables consideradas más relevantes para el modelo. En las figuras Figura 88 y Figura 89 se observa el BurdownChart para las políticas propuestas. En ambas situaciones es similar salvo el segundo caso donde el tiempo total del proyecto es de 223 horas normales.

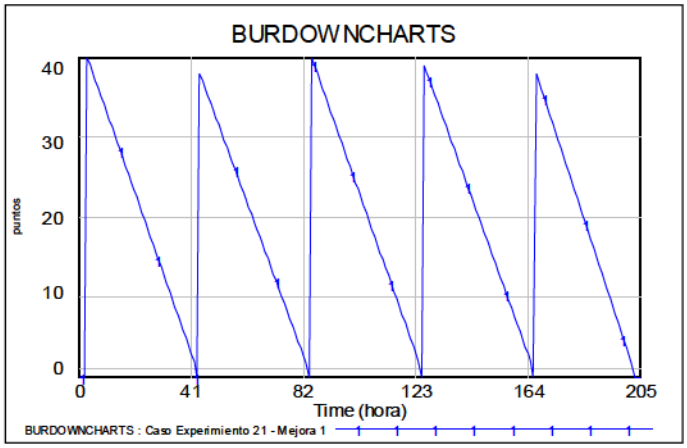


Figura 88 – Experimento 1 – Política 1 – Burdownchart

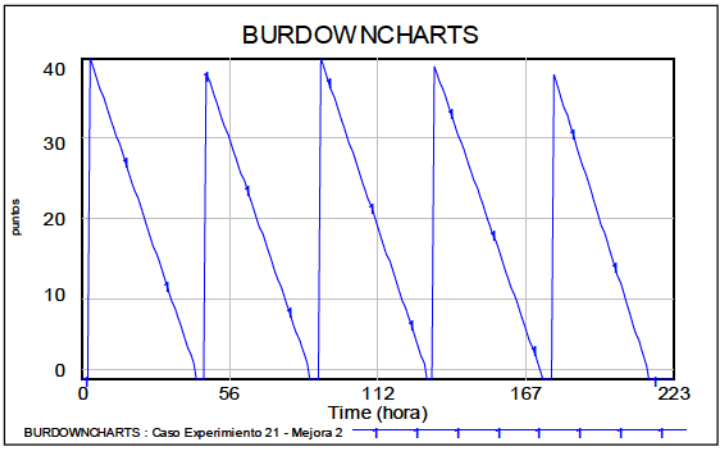


Figura 89 – Experimento 1 – Política 2 – Burdownchart

En la Figura 90 y Figura 91 se observan los resultados de aplicar las mejoras propuestas. En este caso también se incluyen las tareas adicionales.

En la Figura 90 se aprecia que las tareas planificadas son codificadas en su totalidad al final de cada Sprint, quedando un pequeño margen de horas entre la finalización de un Sprint y el inicio de otro. Esto se debe a que el ajuste de velocidad es porcentual y el mismo para todas las variables. En la Figura 91 la totalidad de las tareas son codificadas dado que se adicionó en promedio 2 horas más a cada uno de los Sprints.

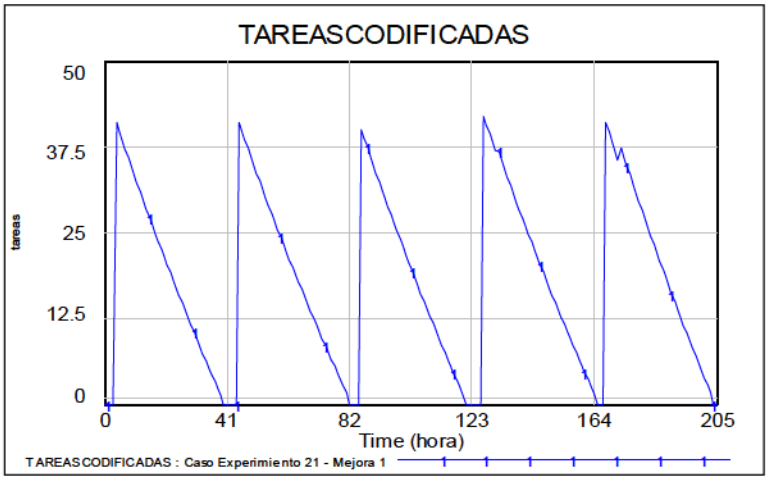


Figura 90 – Experimento 1 – Política 1 –Tareas Codificadas

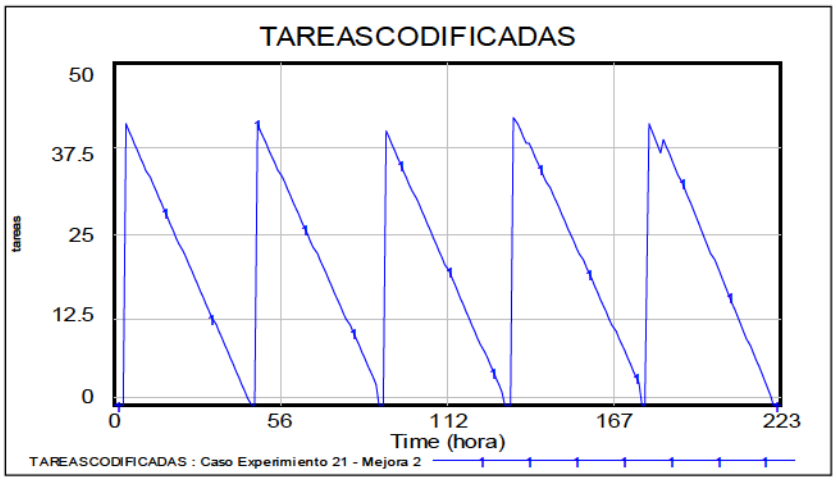


Figura 91 – Experimento 1 – Política 2 – Tareas Codificadas

De igual manera que las tareas codificadas las pruebas a diseñar se completan al momento mismo de la finalización del proyecto. Esto se puede observar la Figura 92 para las políticas 1 y en Figura 93 para las políticas 2.

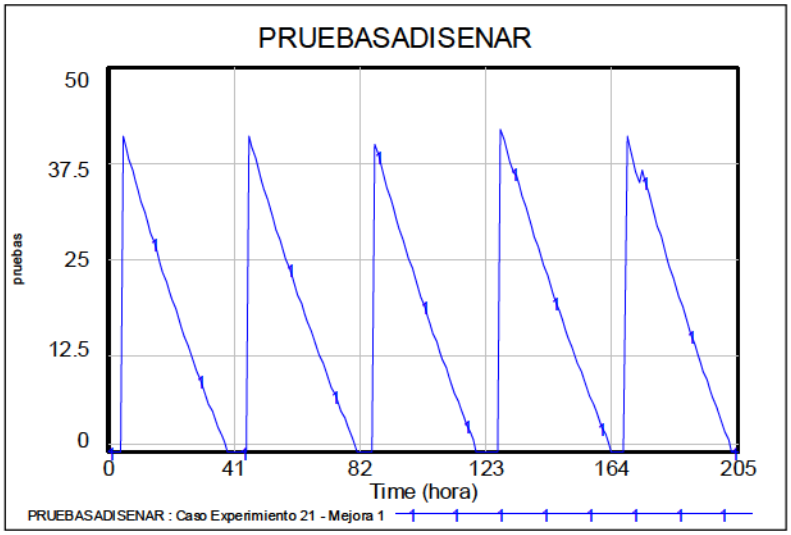


Figura 92 – Experimento 1 – Política 1 – Pruebas a Diseñar

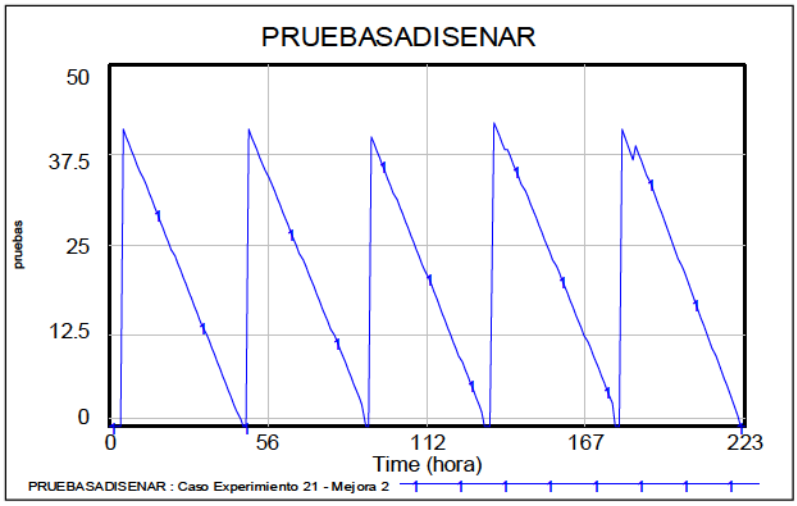


Figura 93 - Experimento 1 – Política 2 – Pruebas a Diseñar

Las diferentes pruebas de integración realizadas, presentan la característica particular de que las tareas con errores no se incluyeron. El punto donde más notoria se hace esta diferencia es en el inicio del último Sprint, donde se puede ver, en la Figura 94 para la política 1 y en la Figura 95 para la política 2, un descenso marcado en las tareas a integrar dada la presencia de errores en las tareas codificadas, lo que lógicamente hace que descienda la cantidad de tareas a integrar.

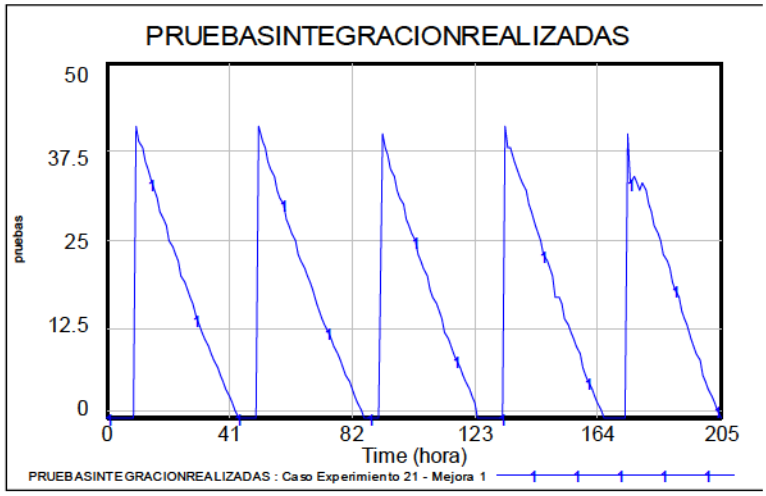


Figura 94 - Experimento 1 – Política 1 – Pruebas de Integración Realizadas

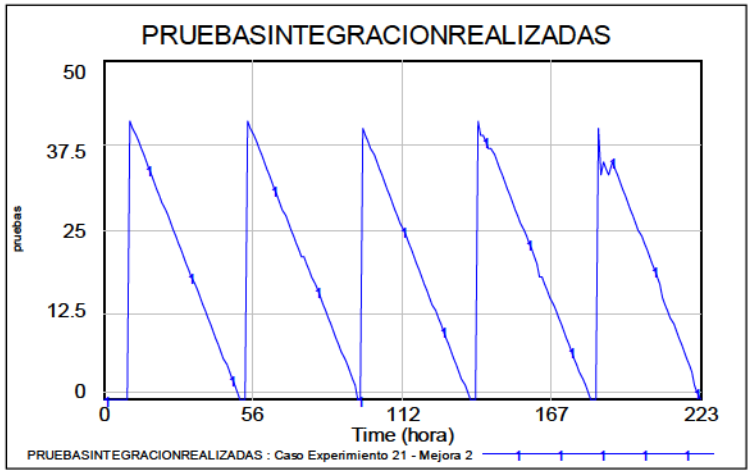


Figura 95 - Experimento 1 – Política 2 –Pruebas de Integración Realizadas

6.2.2.5 Conclusión

Dadas las características del proyecto simulado y los errores detectados, el modelo permitiría al Scrum Master optar por cualquiera de las políticas alternativas propuestas, ya que ambas permiten concluir con la totalidad de lo planificado inicialmente. Independientemente de estas propuestas el Scrum Master podría plantear en los diferentes Sprint Reviews otras mejoras que considere oportunas dada la flexibilidad del modelo al momento de modificar parámetros tales como la velocidad de trabajo diario o la cantidad de Puntos por Sprint. En este caso particular la política 1 sería la más indicada ya que tiene el menor tiempo de duración del proyecto.

6.2.3 Caso de Experimentación 2

En este caso de evaluación del modelo se consideran situaciones donde ciertos parámetros considerados críticos toman valores diferentes a fin de probar el modelo ante situaciones extremas.

En este caso también se considera la Experiencia del Team al momento del cálculo que ejerce la presión en el plazo sobre el Team al momento de desarrollo de tareas.

El Team tiene una formación inicial de 6 integrantes (4 Juniors Promocionales y 2 Juniors Contratados), sobre un total de 8 integrantes requeridos para el proyecto, además se prevé la promoción de un 50% de los Juniors Contratados a Promocionales al inicio del Tercer Sprint. En función de que existe una diferencia inicial entre los integrantes Ideales y los existentes el Team se completará en su totalidad con integrantes Juniors.

6.2.3.1 Parámetros Generales

La Tabla 9 presenta los parámetros generales para el experimento.

Tabla 9 – Experimento 2 - Parámetros Generales

Variable	Valor	Observación
Horas Diarias	7	
Días semana	5	
Horas. Extra por Semana	1	
HorasTotalesporSemana	35	
Horas Totales Normales	210	
Horas Totales Extras	6	
Horas Totales Proyecto	216	
Sprints	6	
Puntos de Historia	214	
PruebasPorTarea	2	
Velocidad Prueba	0	
Inasistencias Pactadas	3	
Inasistencias Imprevistas	5	15%
Factor Dedicación Sprint	70,0%	
Factor Dedicación Diario	80,0%	
Días Hombre Ideal	105	
Días Hombre Real	94,5	
Team	6	
Errores por Presión en el plazo	Si	
Errores de Integración	Si	15%
Promociones en el Team	No	
Abandonos en el Team	No	
Tareas Extras	Si	
Tareas Extras No planificadas	Si	20%

6.2.3.2 Parámetros Iniciales Experimento

La Tabla 10 presenta los valores iniciales utilizados para la corrida.

Tabla 10 – Experimento 2- Parámetros por Sprint

Sprint	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
1	36	1,000	35	0	1	36	2	38	1	1	0
2	34	1,000	35	35	1	36	3	37	0	0	36
3	38	1,000	35	70	1	36	1	39	0	1	72
4	35	0,900	35	105	1	36	2	37	0	0	108
5	35	0,900	35	140	1	36	3	38	1	0	144
6	36	0,990	35	175	1	36	2	38	0	1	180
Total	**214**		**210**		**6**	**216**	**13**	**227**	**2**	**3**	

El detalle de cada columna es el siguiente:

a) Puntos Historia: Determinado por el Team.
b) Duración planificada en horas: Surge de multiplicar las horas normales de trabajo por la cantidad de días semanales de trabajo.
c) Velocidad Estimada: Inicialmente surge de la división de (a) sobre (b). Luego se realizan ajustes para generar el BurdownChart
d) Inicio Planificado: Suma acumulada de los diferentes valores de (b).
e) horas extras: Determinadas según la necesidad del Team. Expresa las horas extras semanales.
f) Duración con Horas. Extras: Surge de la suma de (b) más (e).
g) Tareas Extras: Determinadas por el Scrum Master en función de la necesidad del Team.
h) Puntos con Tareas Extras: Surge de sumar (a) más (g).
i) Tareas Extras No Planificadas: Generado Aleatoriamente.
j) Ausencias No acordadas: Generado Aleatoriamente.
k) Inicio Recalculado: Suma acumulada de los diferentes valores de (f).

6.2.3.3 Observaciones

En este caso de experimentación y a fin de poder ejemplificar las situaciones modeladas, se presentan dos escenarios para las corridas, uno donde se incluyen las tareas extras planificadas y las tareas a recodificar, y otro que presenta los datos base iniciales presentados en la Tabla 9 y Tabla 10.

Además se presentan situaciones en las que como consecuencia de la inclusión de tareas se debió recalcular porcentualmente la velocidad de codificación estimada inicialmente.

6.2.3.4 Resultado del experimento

El inicio de los Sprints para este escenario se observan en la Figura 96.

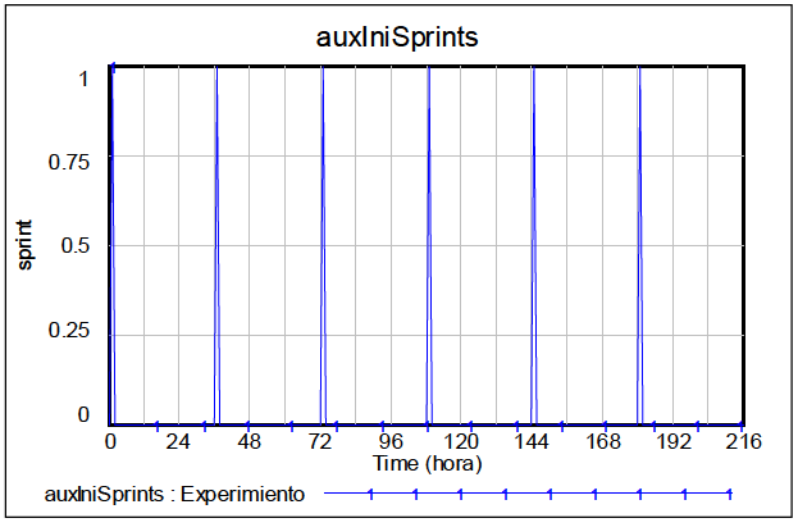


Figura 96 – Experimento 2 - Inicio de los Sprints

Otro de los gráficos relevantes para el modelo presentando es el de los puntos planificados por Sprint, para el actual Experimento dichos puntos se presentan en la Tabla 10, y se muestran en la Figura 97.

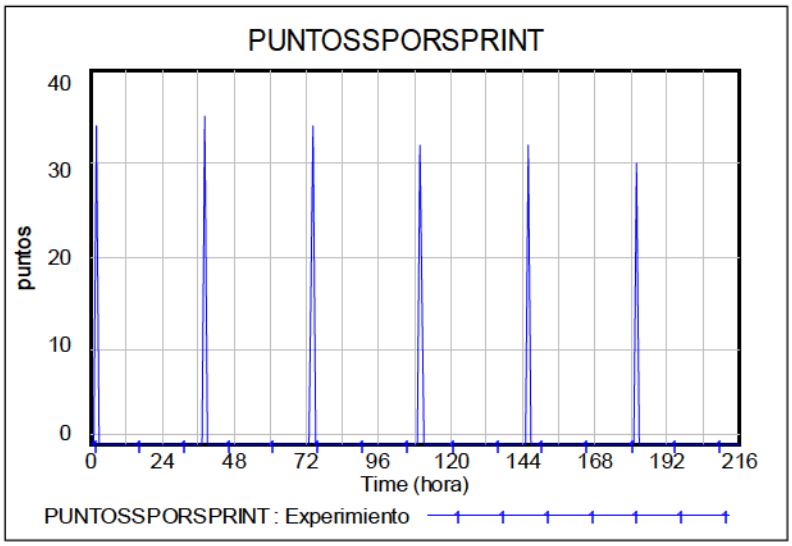


Figura 97– Experimento 2 - Puntos Planificados por Sprint

En función de los puntos planificados y la velocidad estimada de desarrollo se genera el BurdownChart, que muestra un avance ideal del proyecto y del desarrollo de los puntos planificados. En la Figura 98 se observa el comportamiento de dicha variable de nivel.

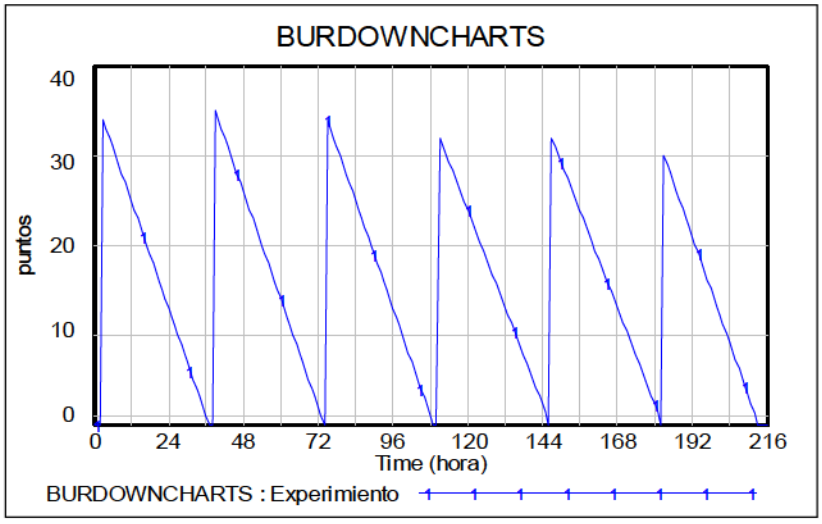


Figura 98– Experimento 2 - BurdownChart

Como se mencionó anteriormente la velocidad de avance permite completar las tareas planificadas y acordadas por el Team. En la Figura 99 se presentan las velocidades para cada uno de los Sprints del presente caso.

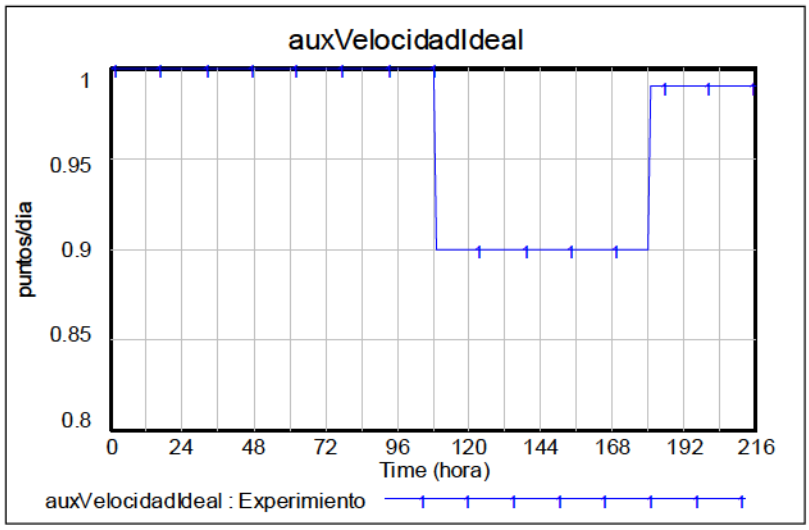


Figura 99– Experimento 2 - Velocidad Ideal Estimada

En función de la velocidad estimada y de los puntos de historia seleccionados para cada sprint, se genera un gráfico que muestra el avance ideal de cómo se deberían ir completando los puntos en cada uno de los Sprints. En la Figura 100 se observa este comportamiento descripto y representado por la variable de nivel PUNTOSPORHACER.

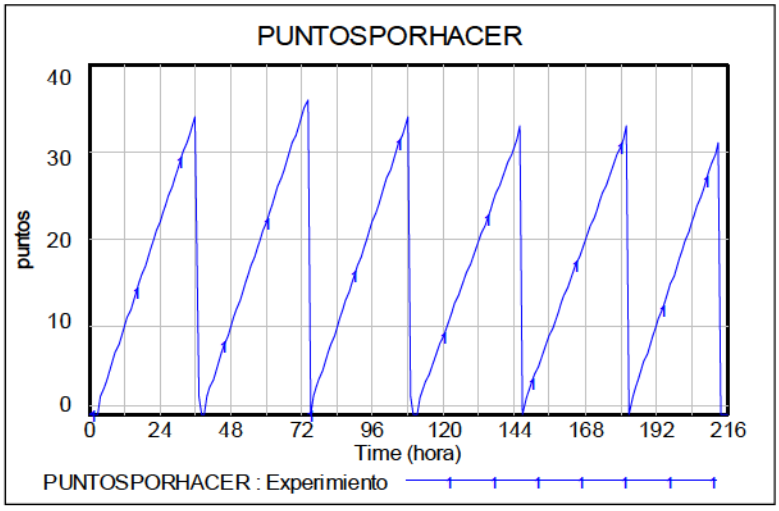


Figura 100– Experimento 2 - Puntos Por Hacer

Manteniendo el criterio inicial de que a cada punto de historia le corresponde una tarea, en la Figura 101 se observa el comportamiento de la variable TAREASPLANIFICADAS que representa el número de tareas según lo acordado inicialmente por el Team.

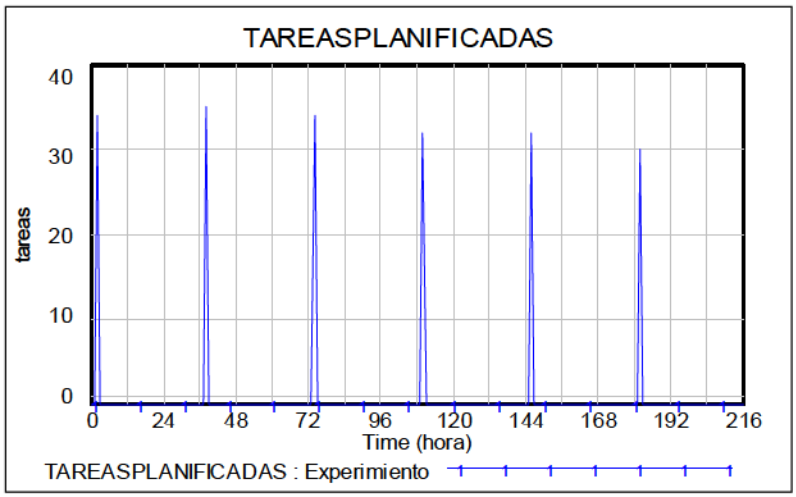


Figura 101 – Experimento 2 - Tareas Planificadas

Como se mencionó al inicio de este Experimento, y como se estimó en los parámetros iniciales de este caso se consideran las tareas que presenten errores de codificación y las tareas extras planificadas a desarrollar. Los valores que estas dos variables se presentan respectivamente en la Figura 102 y Figura 103 respectivamente.

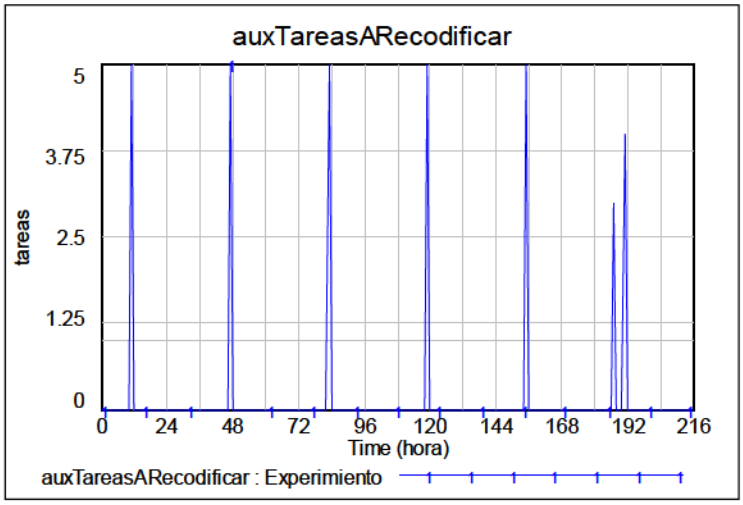


Figura 102 – Experimento 2 - Tareas a Recodificar

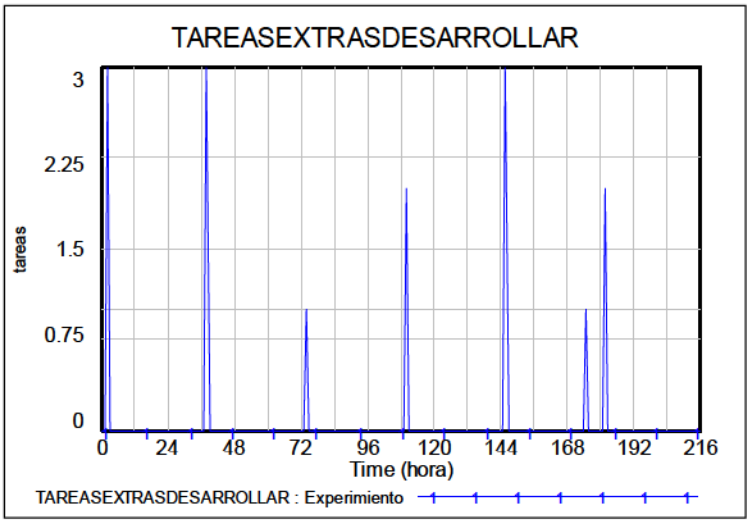


Figura 103 – Experimento 2 - Tareas Extras a Desarrollar

Dada las características particulares de este escenario dónde se presentan tareas extras planificadas y por errores de codificación (Figura 102 y Figura 103) se muestran en la Figura 104 de manera superpuesta las tareas originales, y las tareas originales a las que se adicionaron las tareas extras planificadas. Aunque es sutil la diferencia se aprecia que en la corrida la cantidad de tareas es mayor que en la corrida donde no se incluían las tareas extras.

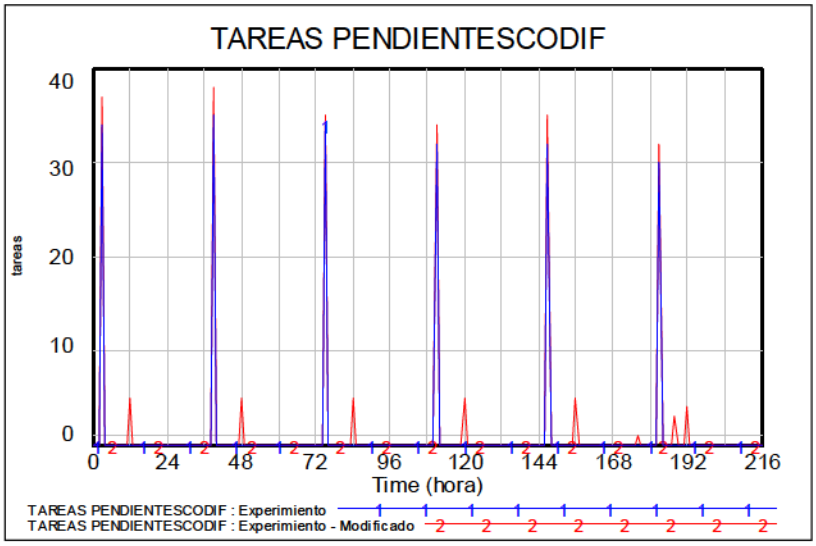


Figura 104 – Experimento 2 - Tareas Pendientes de Codificación

En la Figura 105 se observan de manera superpuesta las corridas donde se incluyeron las Tareas codificadas Con y Sin tareas extras, en este caso se observa que al inicio de cada Sprint la corrida involucra mayor cantidad de tareas, y por lo tanto al final de cada Sprint quedan tareas pendientes por completar, dado que se mantuvo la velocidad estimada inicialmente.

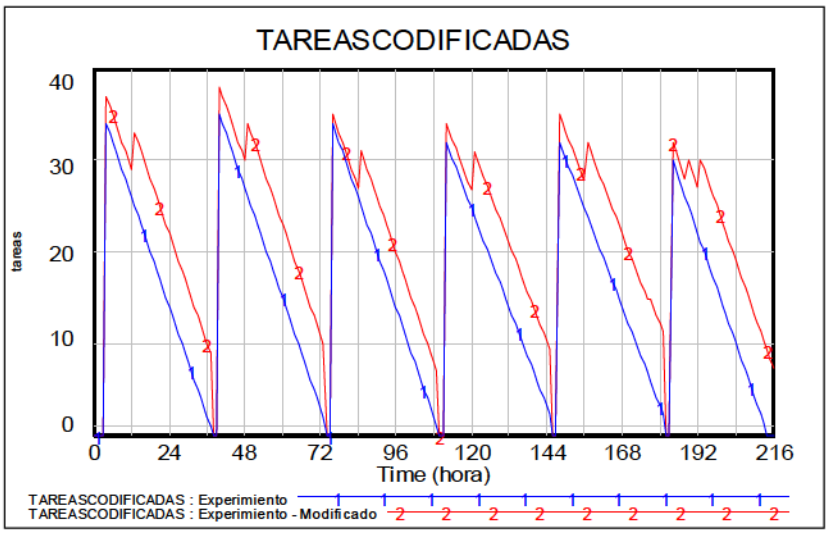


Figura 105 – Experimento 2 - Tareas Codificadas

En la Figura 106 se observa el comportamiento de la variable que representa a las pruebas que deberán ser realizadas sobre las diferentes tareas codificadas.

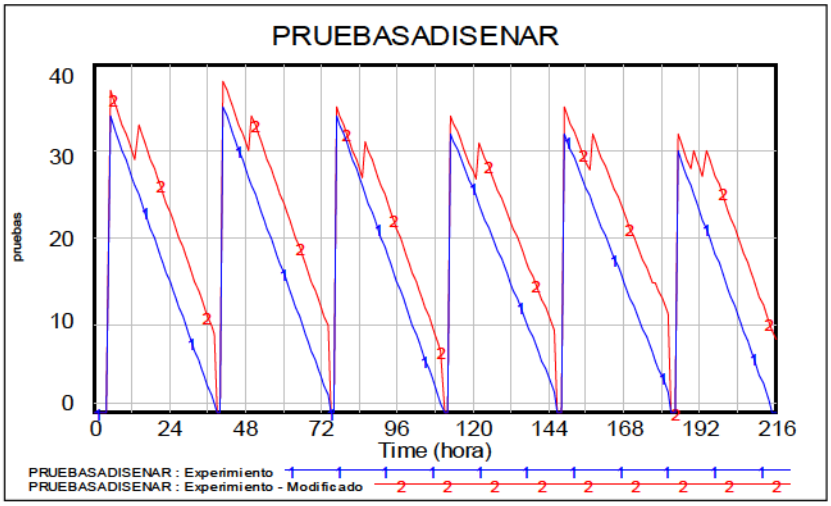


Figura 106– Experimento 2 - Pruebas A Diseñar

Como se mencionó anteriormente en este experimento, se consideran las tareas que por presión en el plazo pudieran sufrir el Team y repercutir en la codificación de las tareas. Es por ello que en la Figura 107 se observan de manera superpuesta el resultado de las diferentes pruebas realizadas sobre las tareas codificadas y aquellas tareas codificadas que no presentaron errores de codificación.

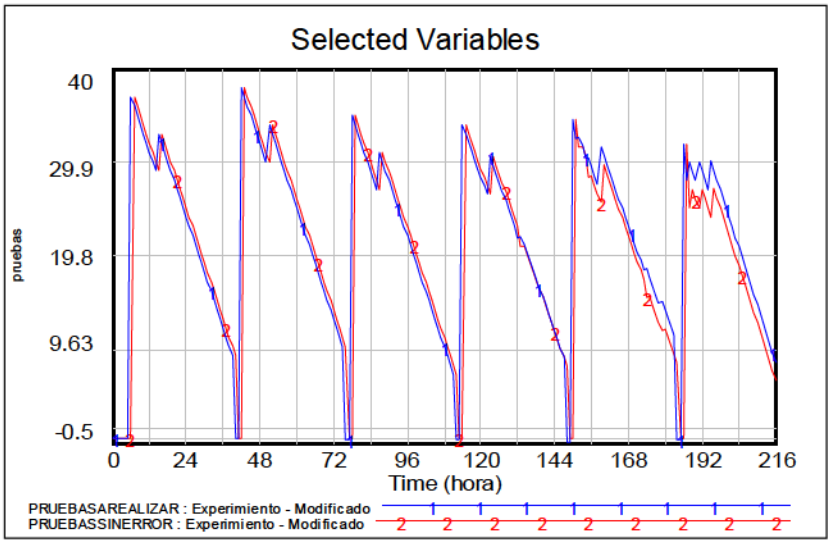


Figura 107– Experimento 2 - Pruebas Sin Errores de Codificación

En la siguiente Figura 108, se presentan las tareas que no presentaron errores de codificación o lógicos, junto a aquellas que si presentaron errores. Si bien es mínima la diferencia, en la figura correspondiente a las tareas sin errores se presentan pequeñas caídas en los momentos donde las tareas con error fueron detectadas.

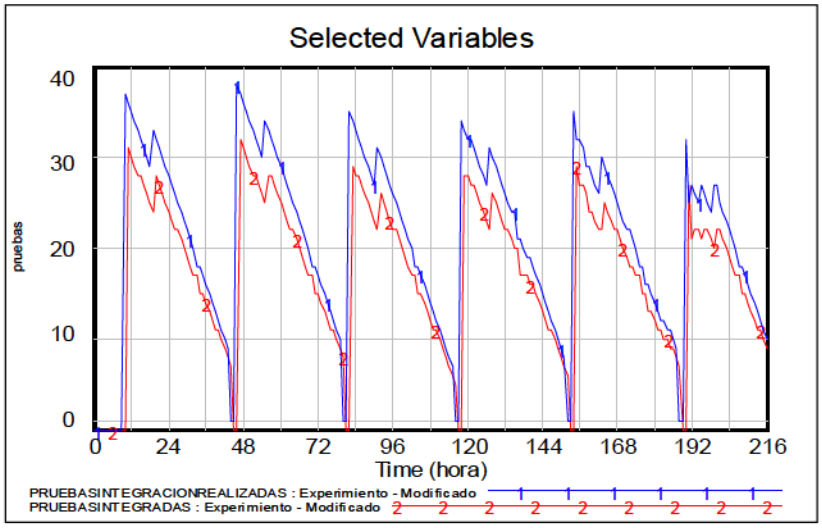


Figura 108– Experimento 2 - Pruebas Con Errores y Sin Errores

6.2.3.5 Políticas Propuestas

En este apartado se presentan dos posibles mejoras frente a la situación que se presentó en el desarrollo del proyecto bajo sus condiciones estimadas inicialmente.

Una de las soluciones es el resultado de realizar una variación en la velocidad diaria de trabajo manteniendo los tiempos iniciales establecidos para la duración de cada Sprint. La otra consiste en la asignación de más horas a la duración de cada Sprint. Ambas soluciones se presentan de manera conjunta a fin de poder realizar una comparación de los resultados obtenidos. En ambos casos los resultados que se obtuvieron corresponden a las corridas que incluían tareas extras.

Las Figura 109, Figura 111, Figura 113 y Figura 115, son el resultado de corregir la situación planteada inicialmente con un ajuste de la velocidad de desarrollo de tareas, logrando de esta manera que todas las tareas planificadas y las tareas no planificadas finalicen en el tiempo estimado inicialmente.

Las Figuras Figura 110, Figura 112, Figura 114 y Figura 116, son el resultado de mantener la velocidad inicial, pero donde a los Sprints se le adicionaron más horas de trabajo

Se observa en la Figura 109 donde se realizó un ajuste de velocidad del 1.5%, quedan pequeños espacios de tiempo sin actividad entre los Sprints dado que el ajuste fue similar en todos los Sprints.

En la Figura 110 se destaca el aumento en la duración del proyecto cuando son adicionadas horas de trabajo a los Sprints, lo que derivaría en un retraso del proyecto y en un aumento en el cansancio del Team.

Ambas figuras presentan además el grafico de BurdownChart a modo de comparación entre lo que se estimó inicialmente y la manera en la que se desarrollaron las tareas planificadas más las tareas extras. Es válido aclarar que si bien aparecen unidades diferentes el análisis y la comparación son válidos ya que a cada punto de historia le corresponde una tarea.

Así en la Figura 109 el desarrollo estimado inicialmente es similar al desarrollo real a lo largo del proyecto. Por su parte en la Figura 110 es notoria la diferencia existente entre lo planificado y lo realizado. Esta última situación llevaría al fracaso del proyecto.

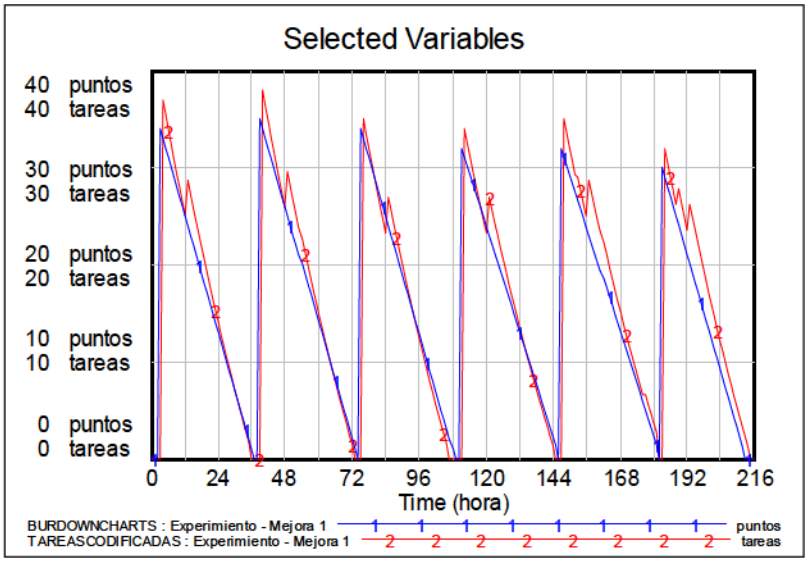


Figura 109 – Experimento 2 – Mejora 1 – Tareas Codificadas

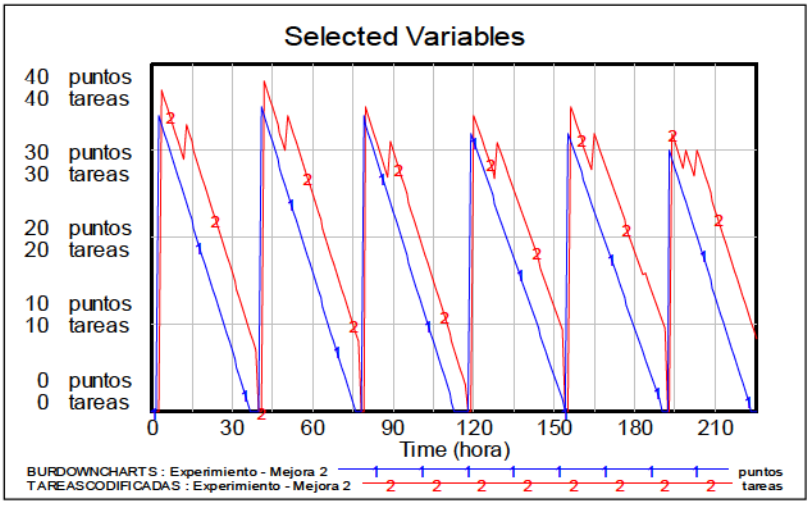


Figura 110 Experimento 2 - Mejora 2 –Tareas Codificadas

Al igual que en el caso de las Tareas Codificadas, las Pruebas a Diseñar también logran completarse en su totalidad en la primera de las mejoras propuestas Pudiendo observarse esto en la Figura 111.

En la Figura 112 se presenta el resultado de adicionar más horas a los Sprints del Experimento haciéndose evidente que no se alcanzaron a realizar todas las pruebas planificadas.

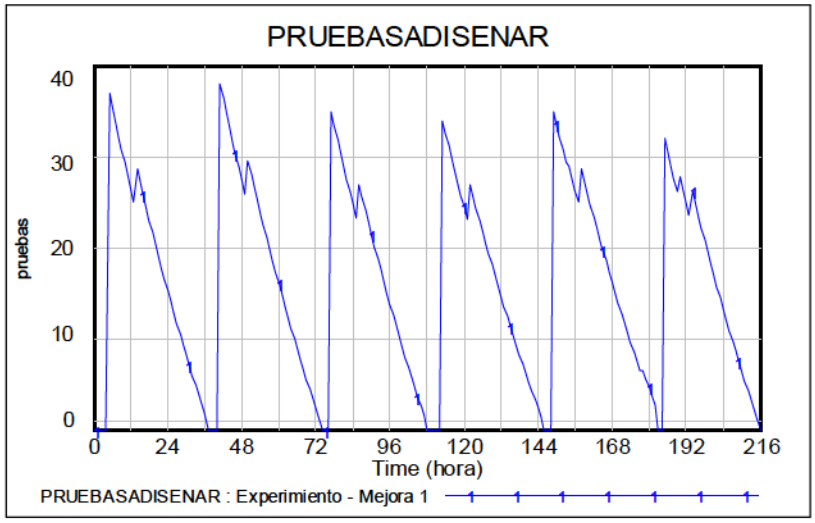


Figura 111 – Experimento 2 - Mejora 1 – Pruebas A Diseñar

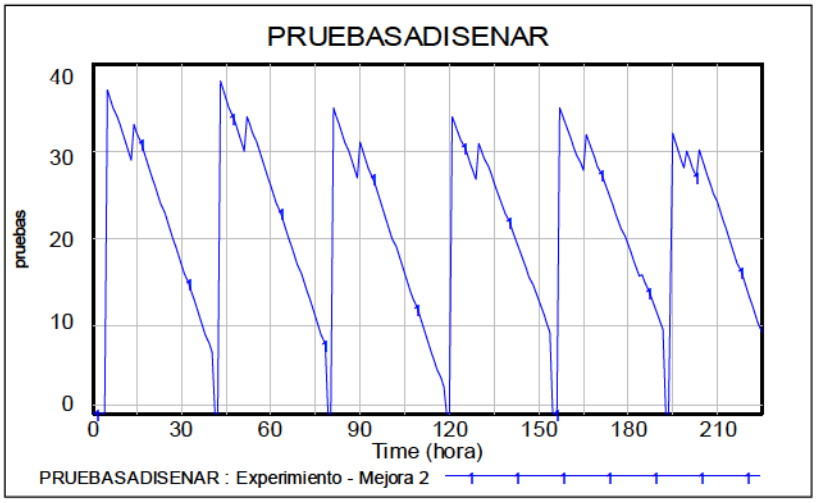


Figura 112 – Experimento 2 - Mejora 2 – Pruebas A Diseñar

En las Figura 113 y Figura 114 se presentan los resultados de las pruebas realizadas y las pruebas que resultaron exitosas, tanto para la mejora donde se realizó un ajuste de velocidad como, en el caso donde se adicionaron horas a los Sprints respectivamente.

Como ocurrió tanto en las Pruebas a Diseñar como en la Codificación de Tareas, en las Pruebas de Integración Realizadas en la primera de las mejoras propuestas se logra completar la totalidad de lo estimado inicialmente. En cambio las pruebas de integración para la segunda mejora propuesta no llegan cumplirse en su totalidad.

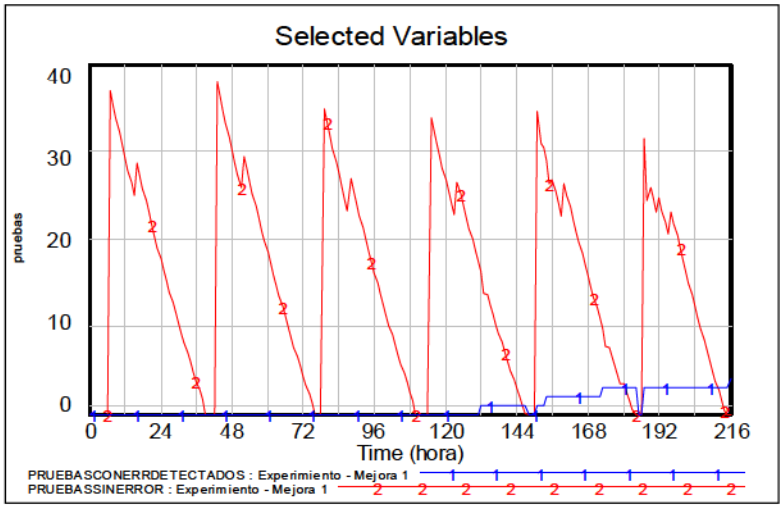


Figura 113 – Experimento 2 - Mejora 1 – Pruebas de Codificación

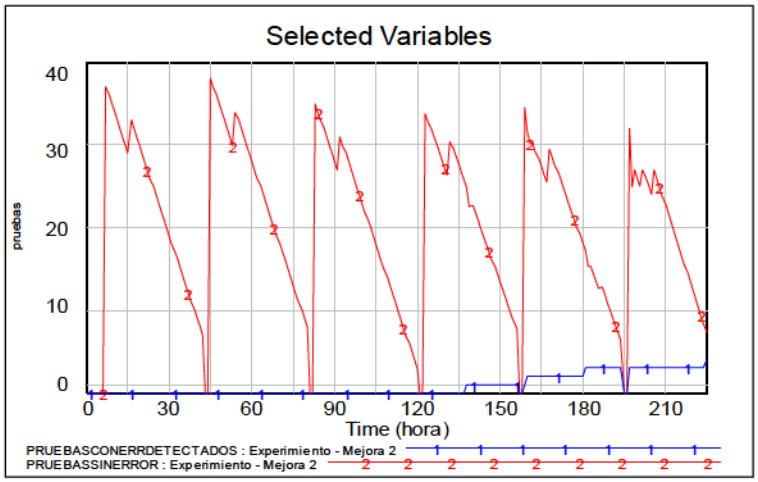


Figura 114– Experimento 2 - Mejora 2 – Pruebas de Codificación

En las Figura 115 y Figura 116 se presentan los resultados de las pruebas de integración realizadas y las pruebas que resultaron exitosas, tanto para la mejora donde se realizó un ajuste de velocidad como, en el caso donde se adicionaron horas a los Sprints respectivamente.

Puede observarse en este caso como para la propuesta de mejora donde se adicionaron horas a los diferentes Sprints se producen espacios entre la finalización y el inicio de un Sprint, dada esta situación podría decirse que el número de horas adicionados fue mayor al necesario, acarreando esto una demora en la entrega del producto final.

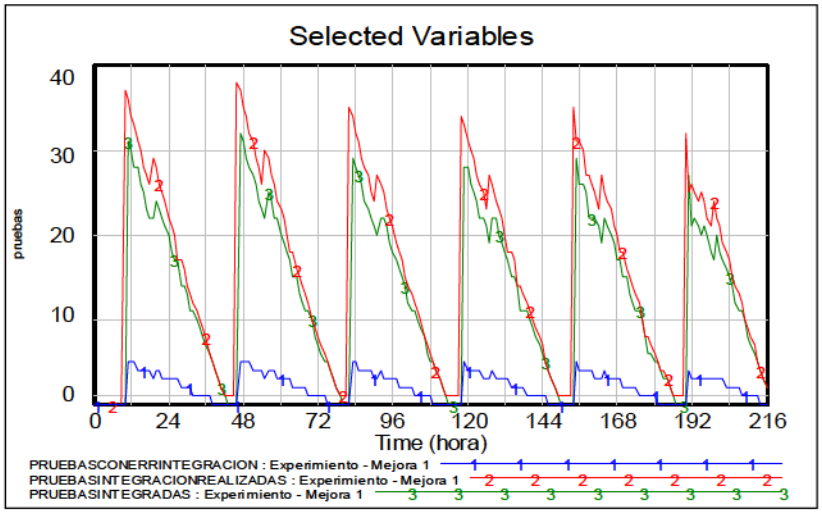


Figura 115 – Experimento 2 - Mejora 1 – Pruebas de Integración

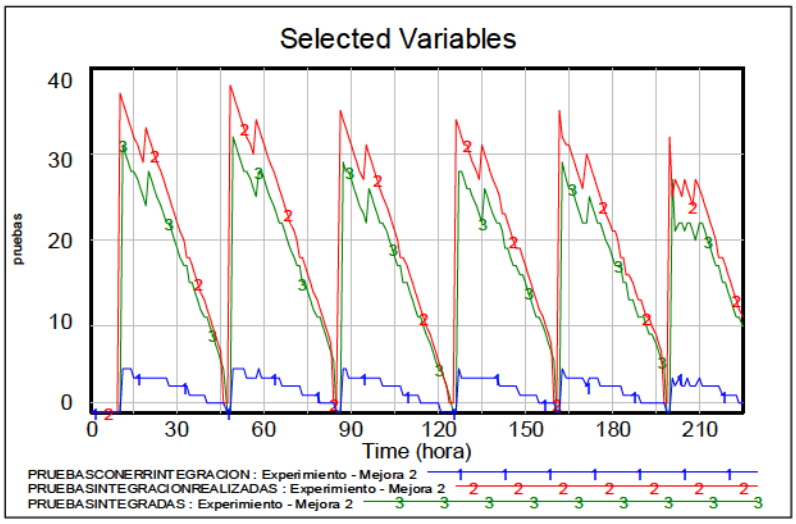


Figura 116 – Experimento 2 - Mejora 2 – Pruebas de Integración

6.2.3.6 Conclusión del experimento.

Como se puede apreciar en este experimento la cantidad excesiva de errores y las ausencias de los integrantes del Team hicieron que el proyecto bajo las condiciones iniciales, presentara un alto número de tareas pendientes de completar al final de los diferentes Sprints, por lo que bajo esas circunstancias el proyecto fracasaría inevitablemente.

Debido a esto, el Scrum Master podría detectar esta situación de modo tal que en el Sprint Planning las plantearía y propondría alguna otra política de gestión del proyecto o bien optar por alguna mejora. En este experimento se optó por un ajuste a la velocidad de trabajo diario y la adición de más horas a los Sprints.

La propuesta de mejora mediante la adición de más horas a los Sprints y en consecuencia al proyecto, no resulta una opción viable ya que aún después de adicionar un promedio de 2 horas a cada Sprint el número de tareas pendientes de desarrollar es elevado. Asímismo, si se agregaran más horas el problema solamente se extendería ya

que la mayor adición de horas traería aparejado una mayor cantidad de errores por cansancio y un atraso mayor en la finalización del proyecto.

Por otro lado, la propuesta donde se realizó un ajuste de la velocidad de trabajo diario resultó la opción más viable dado que se pudieron llevar a cabo todas las tareas planificadas y las diferentes pruebas de control de errores.

6.2.4 Caso de Experimentación 3

Como características principales de este caso de Experimentación se destacan: la adición de una mayor cantidad de ausencias, el porcentaje de dedicación por hora de trabajo es menor, se da una mayor cantidad de tareas extras no planificadas, y la aparición de errores de integración.

Además en este escenario se considera la Experiencia del Team al momento del cálculo que ejerce la presión en el plazo sobre los integrantes, y en relación a la integración del Team. En este caso el Team comienza con 8 integrantes (4 Seniors, 2 Juniors Promocionales, y 2 Juniors Contratados) sobre un total de 8 integrantes requeridos, y así se considera el abandono de un integrante Junior.

6.2.4.1 Parámetros Generales

La Tabla 11 presenta los valores iniciales utilizados para la corrida.

Tabla 11 – Experimento 3 – Parámetros Generales

Variable	Valor	Observación
Horas Diarias	8	
Días semana	5	
Horas.Extra por Semana	0	Variable
Horas Totales por Semana	40	
Horas Totales Normales	240	
Horas Totales Extras	9	
Horas Totales Proyecto	249	
Sprints	6	
Puntos de Historia	214	
Pruebas Por Tarea	1	
Velocidad Prueba	1	
Inasistencias Pactadas	10	
Inasistencias Imprevistas	12	20%
Factor Dedicación Sprint	70,00%	
Factor Dedicación Diario	80,00%	
Días Hombre Ideal	168	
Días Hombre Real	161	
Team	8	
Errores por Presión en el plazo	Si	
Errores de Integración	Si	15%
Promociones en el Team	No	
Abandonos en el Team	Si	
Tareas Extras	Si	
Tareas Extras No planificadas	Si	15%
Búsqueda de nuevos	Si	

6.2.4.1 Parámetros Iniciales del Experimento

La Tabla 12 presenta valores de los parámetros a partir de los cuales se generaron las corridas de este experimento.

Tabla 12 – Experimento 3 – Parámetros Iniciales

Sprint	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
1	36	0,875	40	0	2	42	3	39	0	0	0
2	34	0,825	40	40	2	42	3	37	1	6	42
3	38	0,925	40	80	2	42	3	41	0	1	84
4	35	0,890	40	120	1	41	1	36	1	5	126
5	35	0,890	40	160	1	41	3	38	0	1	167
6	36	1,000	40	200	1	41	5	41	1	0	208
Total	**214**		**240**		**9**	**249**	**18**	**232**	**3**	**12**	

El detalle de cada columna es el siguiente:

a) Puntos Historia: Determinado por el Team.
b) Duración planificada en horas: Surge de multiplicar las horas normales de trabajo por la cantidad de días semanales de trabajo.
c) Velocidad Estimada: Inicialmente surge de la división de (a) sobre (b). Luego se realizan ajustes para generar el BurdownChart
d) Inicio Planificado: Suma acumulada de los diferentes valores de (b).
e) horas extras: Determinadas según la necesidad del Team. Expresa las horas extras semanales.
f) Duración con Horas. Extras: Surge de la suma de (b) más (e).
g) Tareas Extras: Determinadas por el Scrum Master en función de la necesidad del Team.
h) Puntos con Tareas Extras: Surge de sumar (a) más (g).
i) Tareas Extras No Planificadas: Generado Aleatoriamente.
j) Ausencias No acordadas: Generado Aleatoriamente.
k) Inicio Recalculado: Suma acumulada de los diferentes valores de (f).

6.2.4.2 Resultado del Experimento

A fin de hacer más sencilla la lectura y comprensión del caso y todas las posibles variables del mismo que se presentan, a diferencia de los demás experimentos donde se exponía la situación inicial, y luego una serie de propuestas de solución, en este experimento se presentaran los gráficos iniciales correspondientes al Subsistema de Planificación y al Subsistema de Producción, para luego presentar una única alternativa de solución.

Al igual que en los demás casos tanto experimentales como de validación la primera de las Figuras presentadas se corresponde con el momento de inicio de los Sprints para el escenario modelado. En este caso particular dichos inicios se observan en la Figura 117.

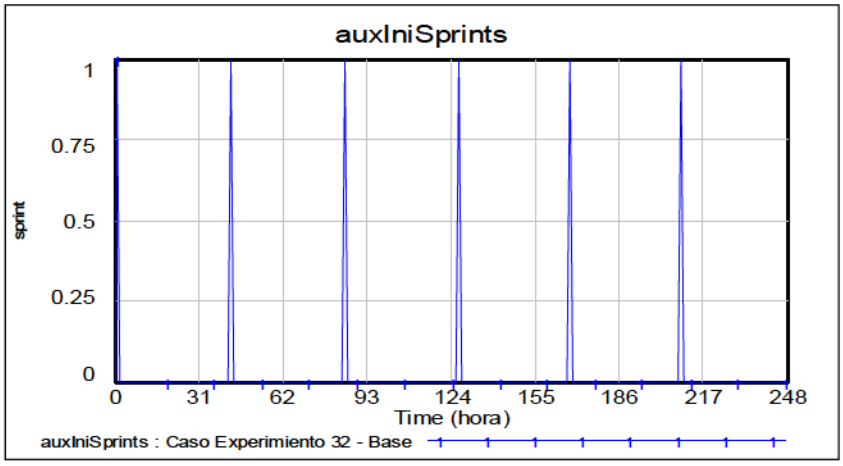


Figura 117 - Experimento 3 – Inicio Sprints

Otro de los gráficos relevantes en el modelo es el que presenta los puntos planificados por Sprint, dado que desde la variable PUNTOSPORSPRINT se generan valores de otras variables del modelo. Para el actual experimento dichos puntos se pueden ver en la Tabla 12, y se muestran en la Figura 118.

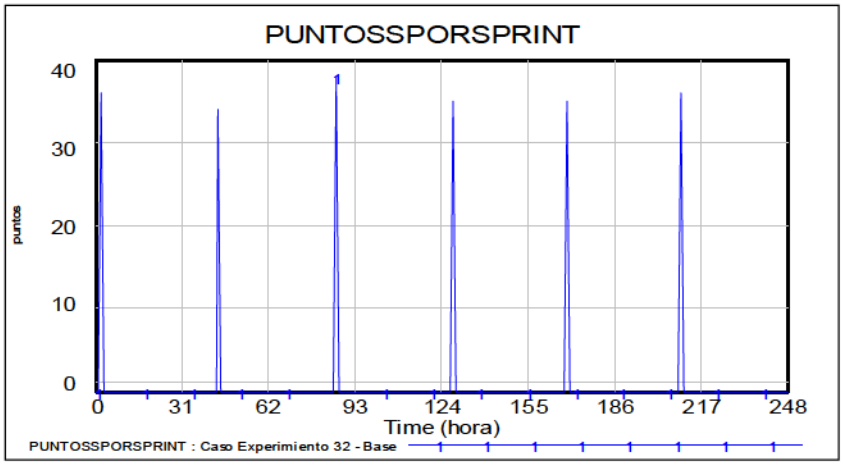


Figura 118 - Experimento 3 – Puntos Por Sprints

En función de los puntos por sprint planificados Figura 118 y a partir de la velocidad estimada inicialmente para el desarrollo de actividades se genera el BurdownChart, que muestra un avance ideal del proyecto y del desarrollo de los puntos planificados. En la Figura 119 se observa el comportamiento de dicha variable de nivel BurdownChart.

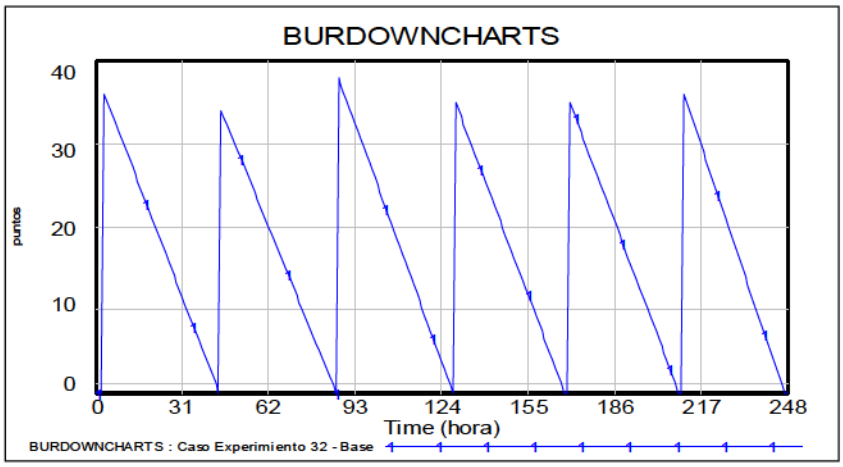


Figura 119 - Experimento 3 – Burdownchart

Como se mencionó a lo largo de los experimentos la velocidad de avance permite completar las tareas planificadas y acordadas por el Team. En la Figura 120 se presenta la velocidad para cada uno de los Sprints del presente experimento. A diferencia de otros casos experimentales, aquí la diferencia de velocidad de los Sprints es más marcada dada la cantidad de puntos seleccionados y la dedicación del Team para desarrollarlos.

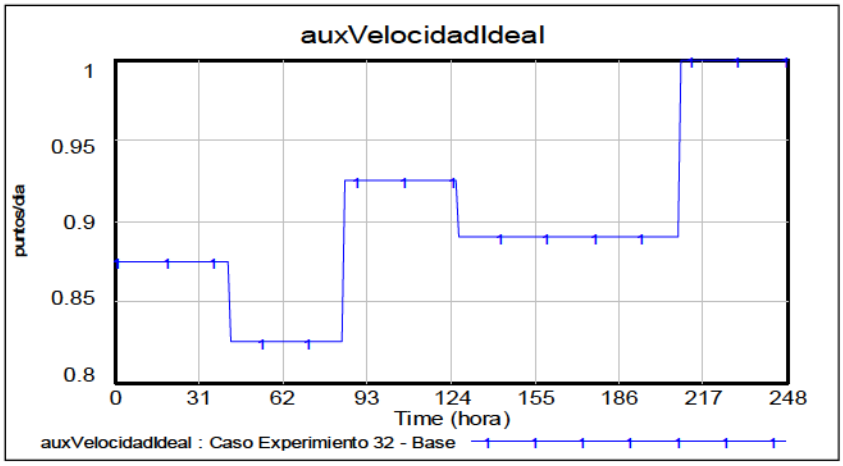


Figura 120 - Experimento 3 – Velocidad Inicia Estimada

En función de la velocidad estimada y de los puntos de historia estimados para cada sprint se genera un grafico que muestra el avance ideal de cómo se deberian ir completando los puntos en cada uno de los Sprints. En la Figura 121 se observa este comportamiento descripto y representado por la variable de nivel PUNTOSPORHACER.

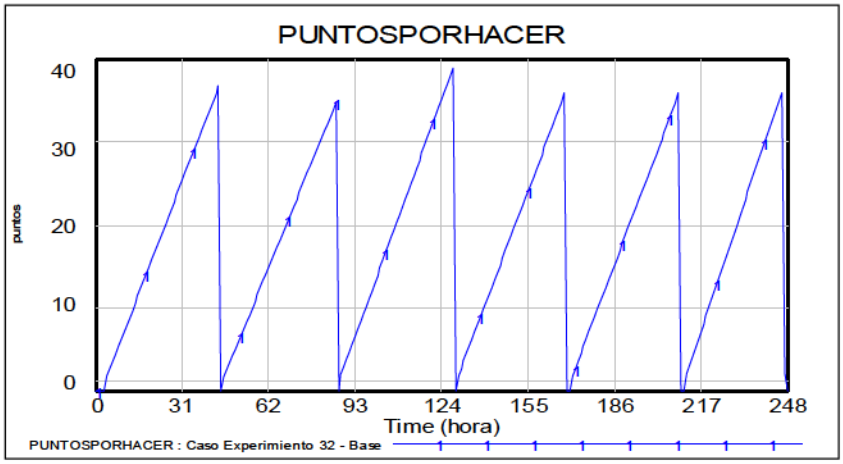


Figura 121 - Experimento 3 – Puntos por Hacer

En función de los puntos planificados y manteniendo siempre el criterio de que a cada punto de historia le corresponde una tarea, en la Figura 122 se observa el comportamiento de la variable TAREASPLANIFICADAS que representa lo acordado inicialmente por el Team.

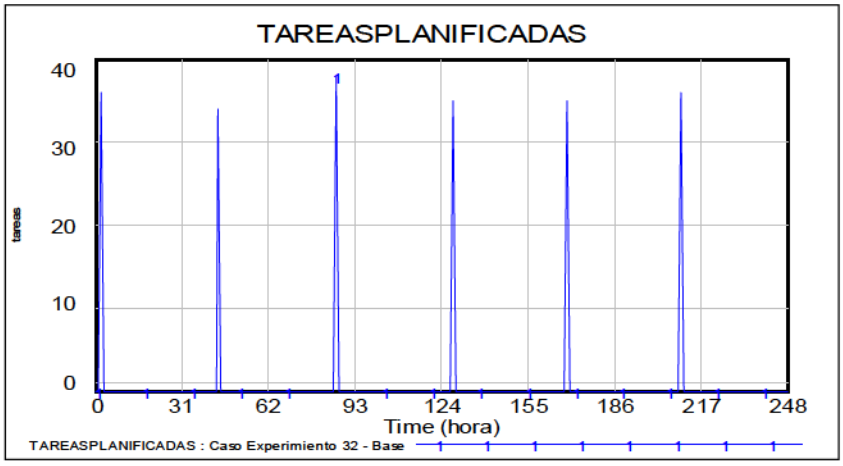


Figura 122 - Experimento 3 – Tareas Planificadas

Como se mencionó al inicio de este apartado en el presente escenario experimental se consideran los errores por presión en el plazo y la probabilidad de errores de integración los asciende al 15% por Sprint. A continuación en la Figura 123 se presenta el grafico correspondiente a las tareas a recodificar únicamente considerando las tareas con errores de integración.

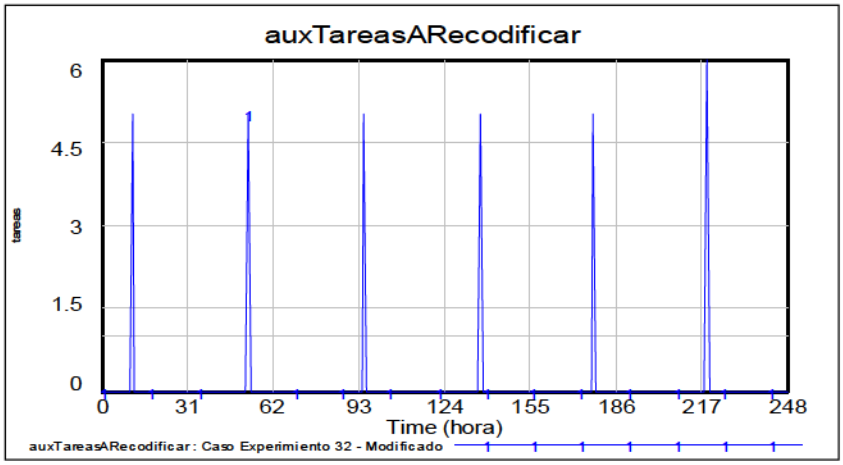


Figura 123 – Experimento 3 – Tareas con Errores de Integración

Como se estimó en los parámetros iniciales de este caso de experimentación se consideran las tareas que presenten errores de codificación por presión en el plazo y de tareas extras planificadas a desarrollar. Los valores que estas dos variables se presentan respectivamente en la Figura 124 y la

Figura 125. Se aprecia que dentro de las tareas a recodificar el número de las mismas es alto, esto se debe a que se consideró para el presente caso un porcentaje elevado de errores de integración, sumado a los errores de codificación por presión en el plazo.

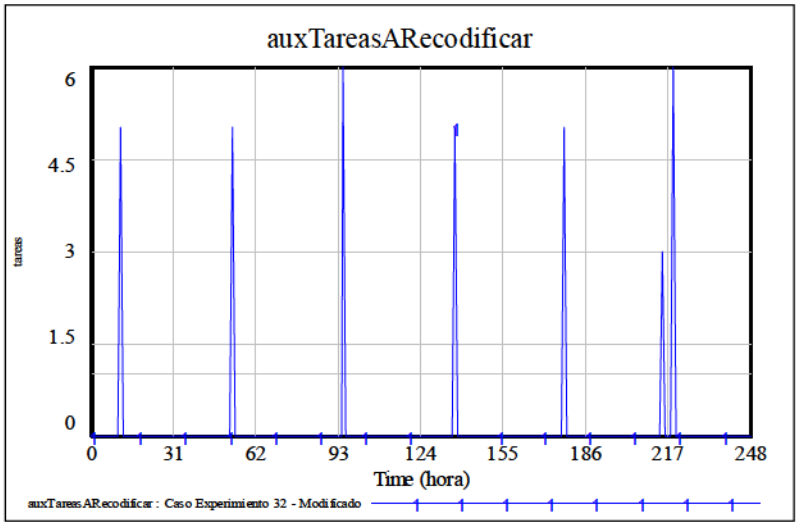


Figura 124 – Experimento 3 – Tareas a Recodificar

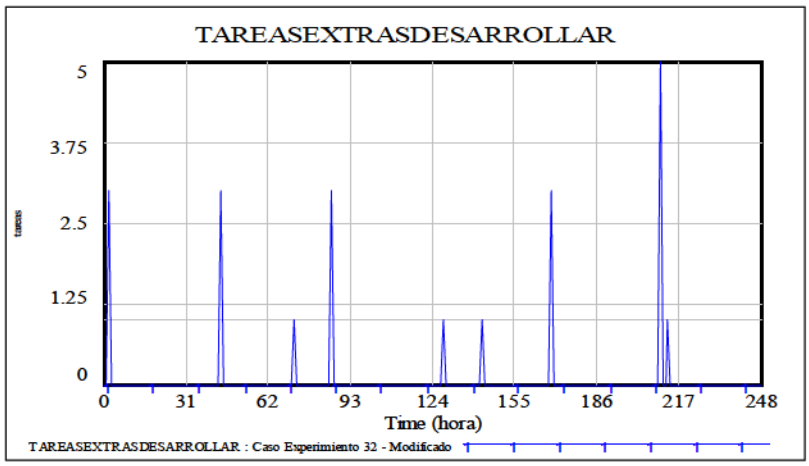


Figura 125 – Experimento 3 – Tareas Extras a Desarrollar

Dada las características particulares de los escenarios de este nivel, y donde se presentan las tareas extras planificadas y por errores de codificación (Figura 124 y Figura 125) se presentan en la Figura 126 las tareas originales a las que se adicionaron las tareas extras planificadas, dando por resultado la totalidad de Tareas Pendientes de Codificación.

Se observa que aparecen nuevas tareas pendientes de codificación en relación al caso inicial, y a diferencia de otros experimentos aparecen pequeños picos correspondientes a las apariciones de tareas no planificadas en diferentes momentos del Sprint.

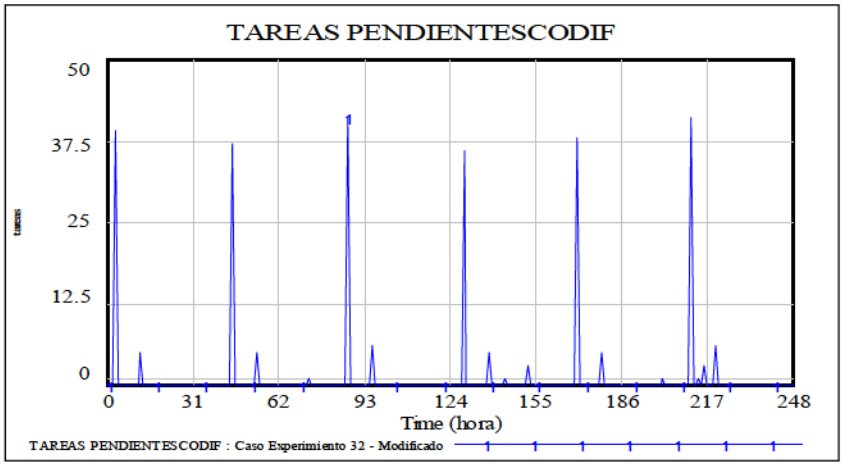


Figura 126 – Experimento 3 – Taras Pendientes de Codificación

Otra de las cuestiones importantes que influyen en el normal desarrollo de las tareas en el proyecto es el nivel de ausencias no acordadas, estas inasistencias se presentan en la Figura 127.

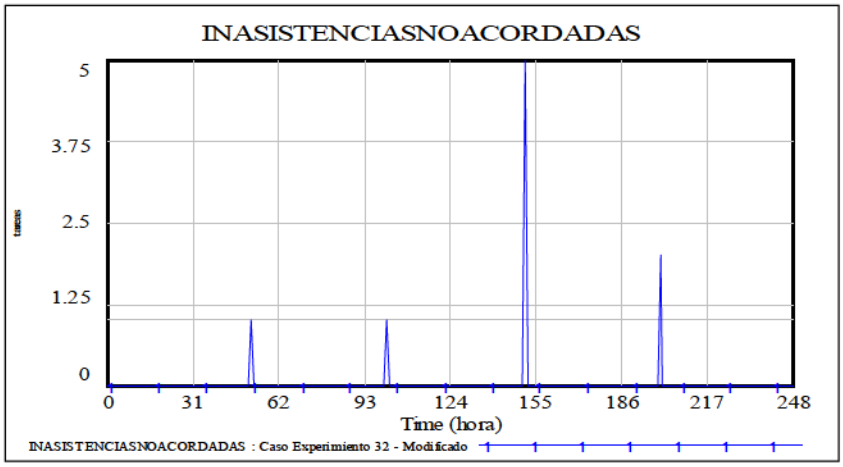


Figura 127 – Experimento 3 – Inasistencias No Acordadas

Considerando la velocidad inicial estimada y la totalidad de tareas pendientes de codificación que se generaron por errores de diferentes tipos más las tareas planificadas, el comportamiento del proyecto se presentaría como se observa en la Figura 128. Es apreciable que de no realizarse algún tipo de ajuste o cambio al final de cada Sprint el número total de tareas pendientes de codificación es alto, rondando aproximadamente 10 tareas por Sprint.

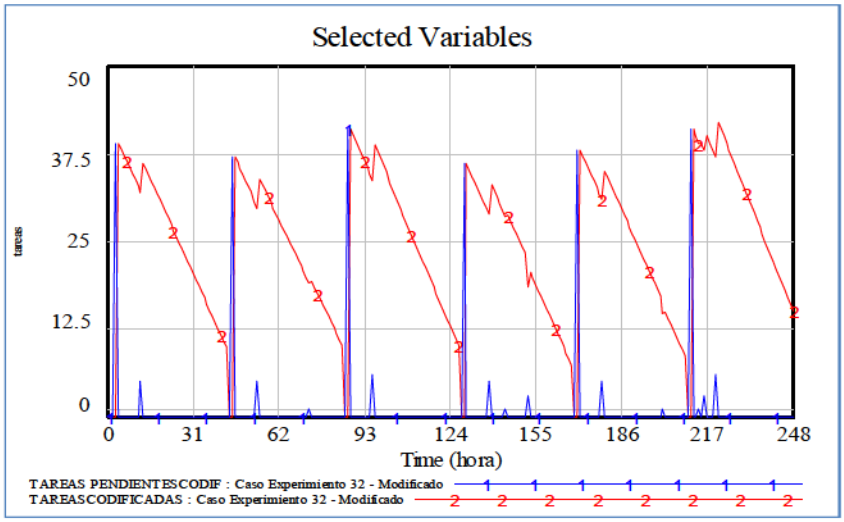


Figura 128 – Experimento 3 – Tareas Codificadas y Tarea Adicionales

En función de lo mencionado en el párrafo anterior y a lo visto en la figura que lo acompañaba, se decide directamente optar por una mejora resultante de la fusión de las propuestas presentadas anteriormente en otros experimentos, optando en este caso por un ajuste de la velocidad de desarrollo de aproximadamente el 3% para todos los Sprints, y una estimación de 8 horas más en el proyecto, llegando a 256 horas. La decisión de esta mejora en forma conjunta se presenta como una alternativa por el hecho de que si solamente se ajustará la velocidad de trabajo sería prácticamente imposible que el Team pudiera realizar la totalidad de lo planificado.

En la Figura 129 se muestra de manera superpuesta las variables que representan las Tareas pendientes de codificación, las tareas codificadas y el BurdownChart. Se observa en la misma figura que el incremento en la velocidad hace que codificación de tareas finalice antes de lo previsto, dado que el Burdownchart fue estimado con una velocidad y las tareas se desarrollaron en base a otra velocidad.

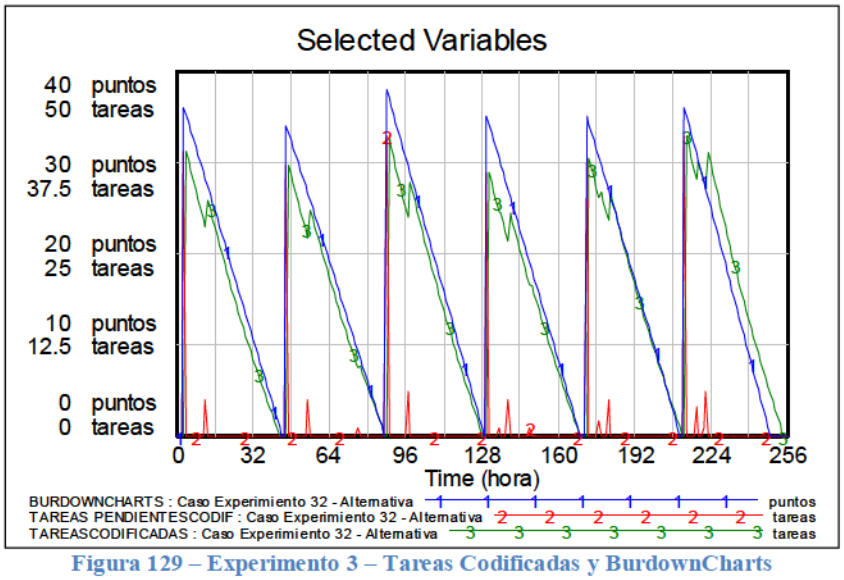


Figura 129 – Experimento 3 – Tareas Codificadas y BurdownCharts

En la Figura 130, se observan las tareas que no presentaron errores de codificación o lógicos debido a la presión(ref.2), junto a aquellas que Si presentaron errores(ref.1). Dada la composicion del Team y la duración del proyecto las tareas con errores no es elevada, quedando esto visible en la Figura 129.

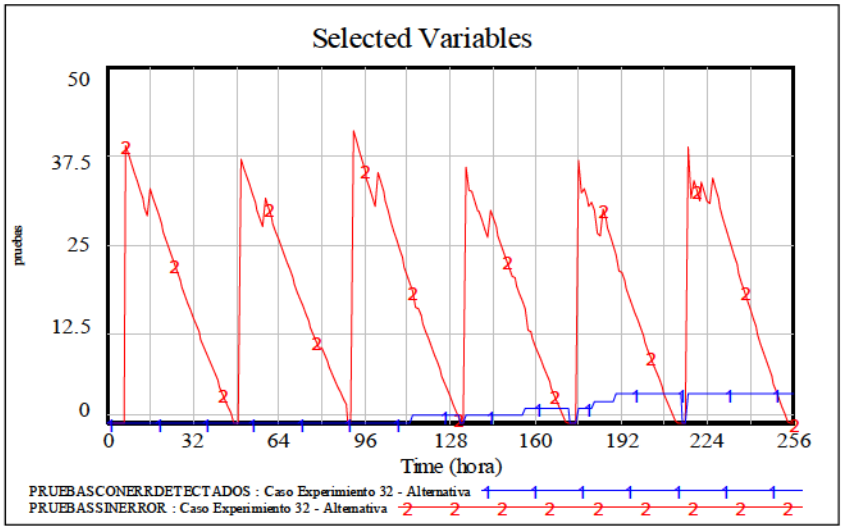


Figura 130 – Experimento 3 – Pruebas de Codificación

Como anteriormente se observó en los Diagramas de Forrestercon las tareas que no presentaron errores por Codificación se procede a la generación de las pruebas de integración con las mismas. Dado que para el presente escenario se consideró una tasa de error del 15% dicha tasa incide en las tareas produciendo el resultado de Figura 131.

Aunque más difícil (dada los valores de otras variables que afectaron a las tareas sin errores), se observa en la figura que existe una diferencia entre las pruebas de integración diseñadas y la cantidad de pruebas sin errores, pero también se observan las tareas que si presentaron errores de integración y se corresponden con dicha diferencia.

Además se aprecia que el resultado del comportamiento de la variable de la Figura 131 se ve afectado en mayor medida por las tareas con errores de integración.

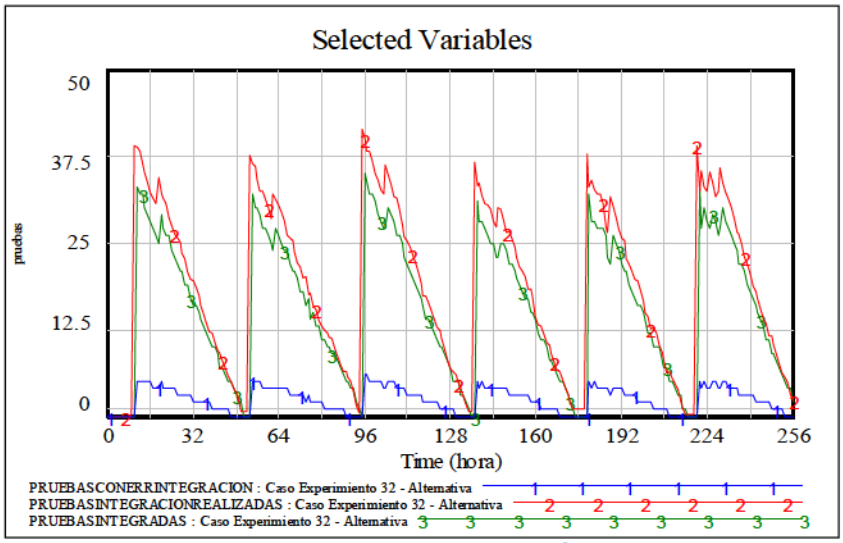


Figura 131– Experimento 3 – Integración de Tareas

Una de las cuestiones más relevantes en este experimento como se mencionó es el abandono de 2 integrantes Juniors y la contratación de Seniors en su reemplazo. Se consideró arbitrariamente que este evento ocurriera a las 80 horas de iniciado el proyecto. El comportamiento de la variable que representa a los Juniors Contratados para este proyecto se observa en la Figura 132.

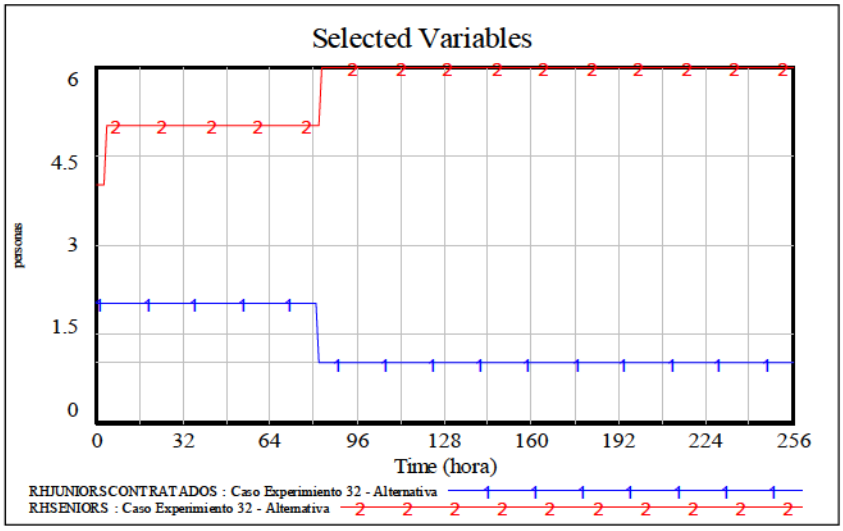


Figura 132 – Experimento 3 – Abandono Juniors

Como el comportamiento de la experiencia depende de la cantidad de integrantes y de la experiencia de estos a continuación en la Figura 133 se presenta la variable que representa la experiencia total del Team, y se ve el momento en el cual el integrante Junior abandona el Team y el nuevo integrante Experto pasa a formar parte del equipo.

Como no se considera ningún retardo entre el abandono del Junior y el ingreso del experto, es por ello que la experiencia del Team mejora pasando de aproximadamente 0.6 puntos de experiencia a casi 0.8 puntos.

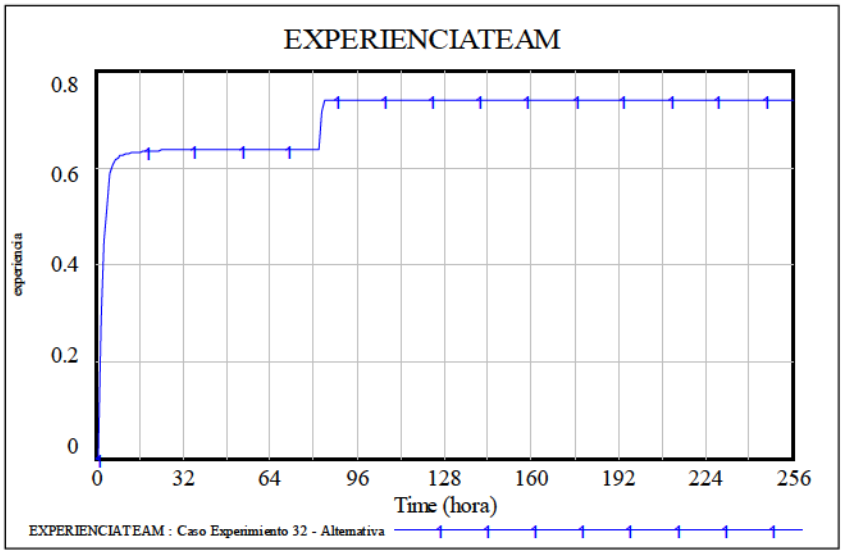


Figura 133 - Experimento 3 – Experiencia Team

6.2.4.3 Conclusión

Como se mencionó en el desarrollo de este experimento, se presentó una única alternativa de solución resultante de la fusión de las propuestas presentadas en otros experimentos, consistente en la adición de más horas a los Sprints y de realizar un ajuste a la velocidad de trabajo diario.

La selección de esta mejora podría ser la alternativa elegida por el Scrum Master previo al inicio del proyecto o en un Sprint Review, donde viendo que no se pudo alcanzar el objetivo inicialmente propuesto para un Sprint decide optar por una alternativa como la propuesta en este experimento para los Sprints restantes.

En relación al resultado obtenido mediante la propuesta de mejora resulta en principio una opción viable, ya que después de implementarla el número de tareas pendientes de desarrollo se completa al igual que las diferentes pruebas para la detección de errores de lógica o de codificación. Pero como se observa en la Figura 131 el problema se presenta al momento de realizar pruebas de integración, donde en el último Sprint no se llega a completar la totalidad de las pruebas de integración planificadas.

Otra de las características se da en la composición y la experiencia del Team, donde frente ante el abandono de Juniors y la incorporación de integrantes Seniors el nivel de experiencia alcanzado no permitió que se pudieran desarrollar el total de las tareas planificadas, es por esto que la alternativa de mejora propuesta para este caso no es suficiente.

6.2.5 Caso de Experimentación 4

A continuación se presenta el tercero de los experimentos considerados complejos. Dentro de los cambios más relevantes en este escenario en relación a los anteriores es que se contempla la incorporación de nuevos integrantes al Team.

En relación a los experimentos anteriores este caso experimental se diferencia de aquellos básicamente en que se incluye la adición de una mayor cantidad de ausencias, la incorporación de nuevos integrantes Juniors al Team.

Empieza a considerarse la Experiencia del Team al momento del cálculo que ejerce la presión en el plazo sobre el Team al momento de desarrollo de tareas.

El Team tiene una formación inicial de 6 integrantes (4 Juniors Promocionales y 2 Juniors Contratados), sobre un total de 8 integrantes requeridos para el proyecto. En función de que existe una diferencia inicial entre los integrantes Ideales y los existentes el Team se completará en su totalidad con integrantes Juniors.

6.2.5.1 Parámetros Generales

La Tabla 13 presenta los valores de los parámetros generales, establecidos por el scrum master y el Team, que permiten calcular los valores de la Tabla 14, y parámetros por sprint para utilizados en este experimento. Esta es la situación base.

Tabla 13 – Experimento 4 – Parámetros Generales

Variable	Valor	Observación
Horas Diarias	7	
Días semana	5	
Horas.Extra por Semana	0	variable
Horas Totales por Semana	35	
Horas Totales Normales	245	
Horas Totales Extras	14	
Horas Totales Proyecto	259	
Sprints	7	
Puntos de Historia	214	
Pruebas Por Tarea	1	
Velocidad Prueba	1	
Inasistencias Pactadas	15	
Inasistencias Imprevistas	18	15%

Factor Dedicación Sprint	80,0%	
Factor Dedicación Diario	70,0%	
Días Hombre Ideal	240	
Días Hombre Real	228	
Team Ideal	8	
Team Inicial	6	
Errores por Presión en el plazo	Si	
Errores de Integración	Si	15%
Promociones en el Team	No	
Abandonos en el Team	No	
Tareas Extras	Si	
Tareas Extras No planificadas	Si	15%
Búsqueda de nuevos integrantes	Si	Únicamente Juniors

6.2.5.2 Parámetros Iniciales Experimento

Tabla 14 – Experimento 4 – Parámetros por Sprint

Sprint	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)
1	29	0,810	35	0	2	37	3	32	2	0	0
2	31	0,850	35	35	2	37	3	34	4	4	37
3	31	0,850	35	70	2	37	3	34	1	5	74
4	34	0,950	35	105	2	37	1	35	3	0	111
5	29	0,800	35	140	2	37	3	32	3	3	148
6	31	0,850	35	175	2	37	5	36	6	4	185
7	29	0,870	35	210	2	37	5	34	5	2	222
	214		**245**		**14**	**259**	**23**	**237**	**24**	**18**	

El detalle de cada columna es el siguiente:

a) Puntos Historia: Determinado por el Team.
b) Duración planificada en horas: Surge de multiplicar las horas normales de trabajo por la cantidad de días semanales de trabajo.
c) Velocidad Estimada: Inicialmente surge de la división de (a) sobre (b). Luego se realizan ajustes para generar el BurdownChart
d) Inicio Planificado: Suma acumulada de los diferentes valores de (b).
e) horas extras: Determinadas según la necesidad del Team. Expresa las horas extras semanales.
f) Duración con Horas. Extras: Surge de la suma de (b) más (e).
g) Tareas Extras: Determinadas por el Scrum Master en función de la necesidad del Team.
h) Puntos con Tareas Extras: Surge de sumar (a) más (g).
i) Tareas Extras No Planificadas: Generado Aleatoriamente.
j) Ausencias No acordadas: Generado Aleatoriamente.
k) Inicio Recalculado: Suma acumulada de los diferentes valores de (f).

6.2.5.3 Resultado del experimento

La primera Figura de este experimento corresponde a la variable auxiliar que indica el inicio de cada uno de los Sprints planificados. El momento en el que se inician los Sprints se basan en la Tabla 13 y se reflejan en la Figura 134.

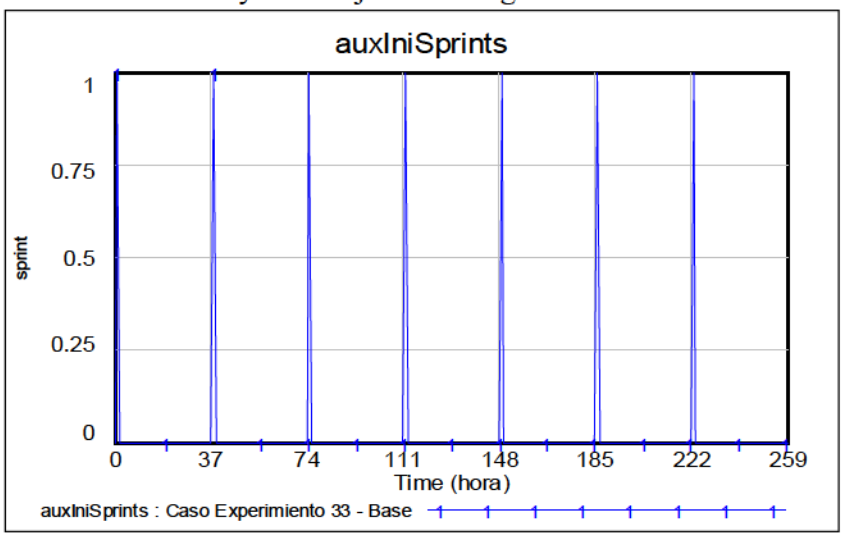


Figura 134 – Experimento 4 – Inicio Sprints

Otra de las variables relevantes en el experimento corresponde a los puntos de historia que deben ser completados en los diferentes Sprints. La Figura 135 presenta estos puntos.

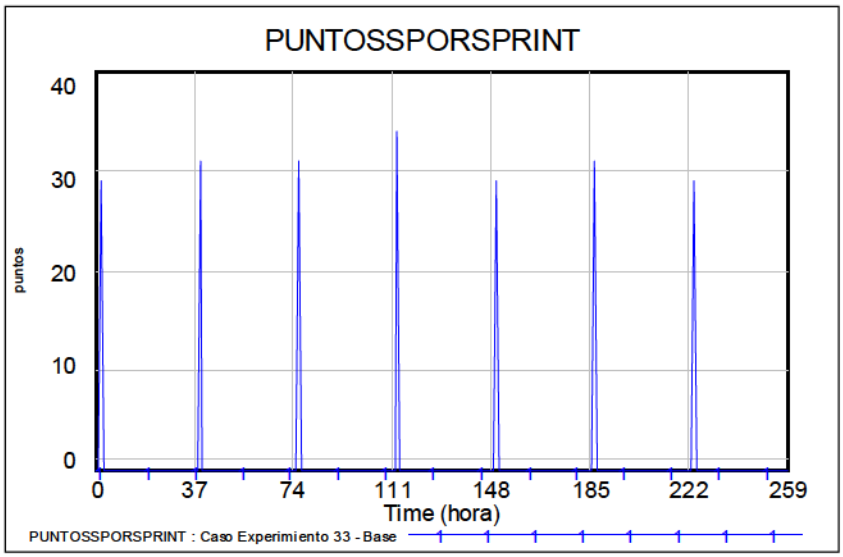


Figura 135 – Experimento 4 – Puntos Planificados Por Sprint

En función de los puntos planificados y de velocidad estimada de desarrollo se genera el BurdownChart, que muestra un avance ideal del proyecto y del desarrollo de los puntos planificados. En la Figura 136 se observa el comportamiento de dicha variable de nivel

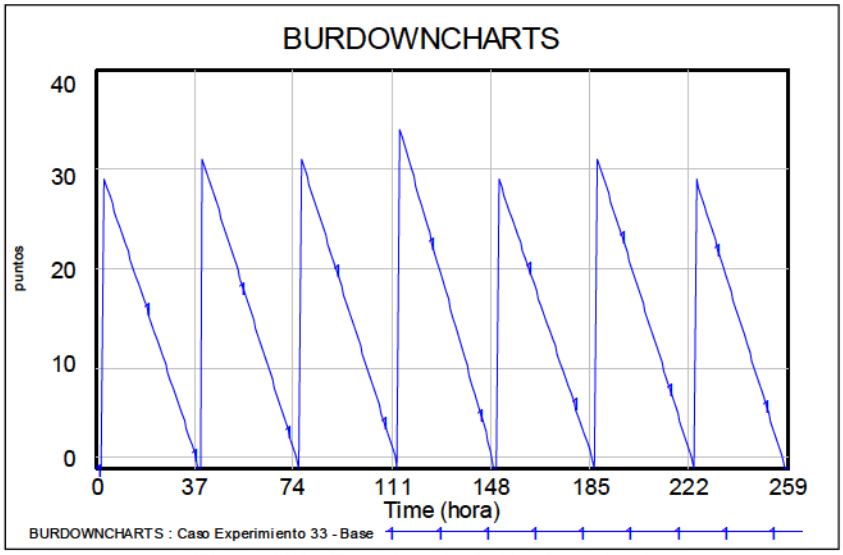


Figura 136 – Experimento 4 –Brudownchart

La velocidad inicial estimada de avance permite completar las tareas planificadas y acordadas en el Team. En la Figura 137 se presenta la velocidad para cada uno de los Sprints del presente caso.

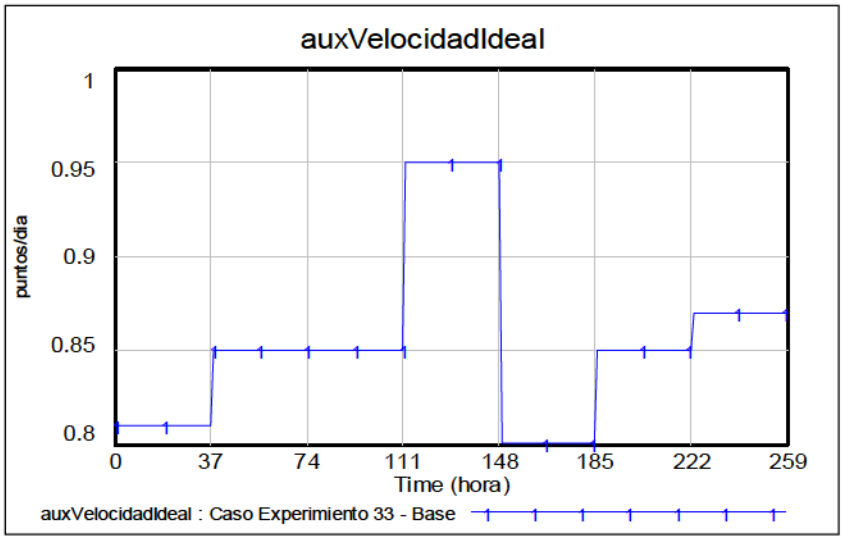


Figura 137 – Experimento 4 – Velocidad Ideal Estimada

En función de la velocidad estimada y de los puntos de historia seleccionados para cada sprint se genera un grafico que muestra el avance ideal de cómo se deberian ir completando los puntos en cada uno de los Sprints. En la Figura 138 se observa este comportamiento descripto y representado por la variable de nivel PUNTOSPORHACER.

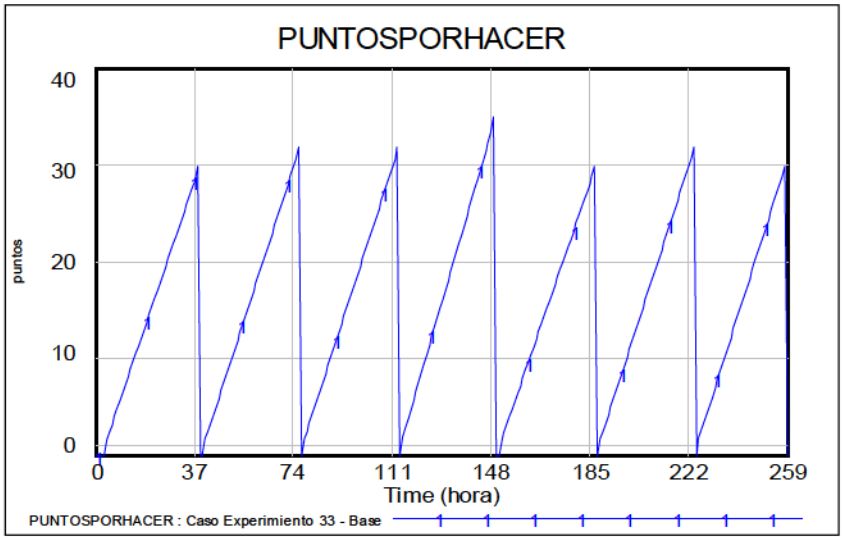


Figura 138 – Experimento 4 – Puntos Por Hacer

Como se considera que a cada punto de historia le corresponde una tarea en la Figura 139 se observa el comportamiento de la variable TAREASPLANIFICADAS.

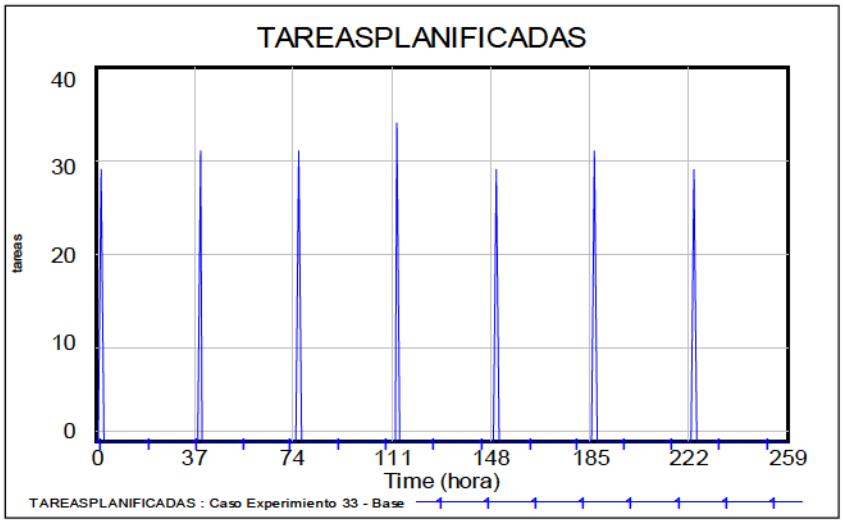


Figura 139 – Experimento 4 – Tareas Planificadas Iniciales

Siendo este uno de los experimentos más complejos y en el cual entran en juego el mayor número de variables del modelo, la cantidad de tareas que se deben recodificar difiere significativamente de los demás experimentos.

En la Figura 140 se observan las diferentes tareas que presentaron algún tipo de error, ya sea por la Presión en el Plazo, por errores de Integración o bien tareas extras que debe desarrollar el Team. En la

Figura 141 se muestran las diferentes Tareas extras planificadas por el Team.

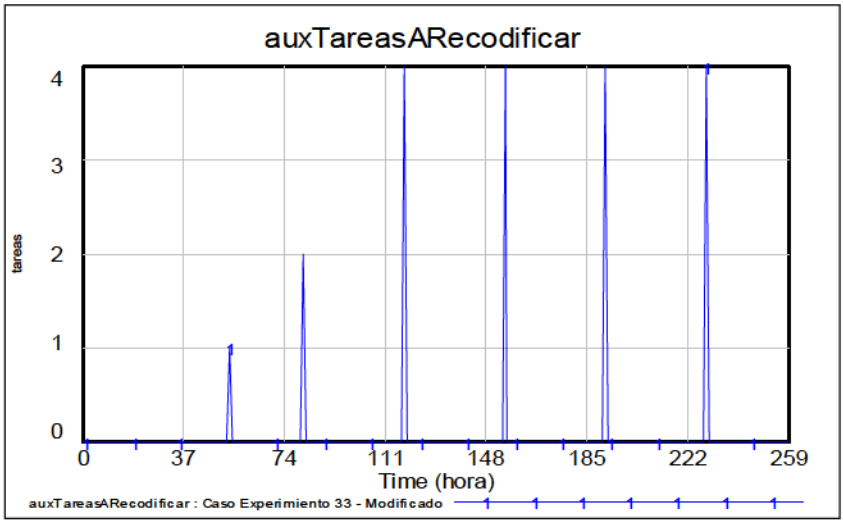


Figura 140 – Experimento 4 –Tareas a Recodificar

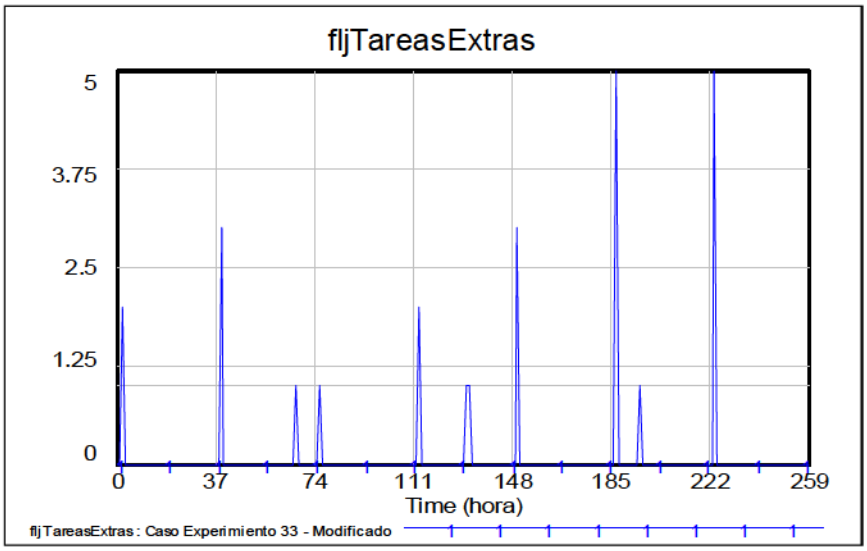


Figura 141 – Experimento 4 –Tareas Extras Planificadas

Como se mencionó y dado el número elevado de las diversas tareas a recodificar y tareas extras que deben desarrollar se genera el conjunto de Tareas pendientes de codificación. La Figura 142, es el resultado de la suma de las diversas tareas a recodificar, las tareas extras y las tareas planificadas inicialmente.

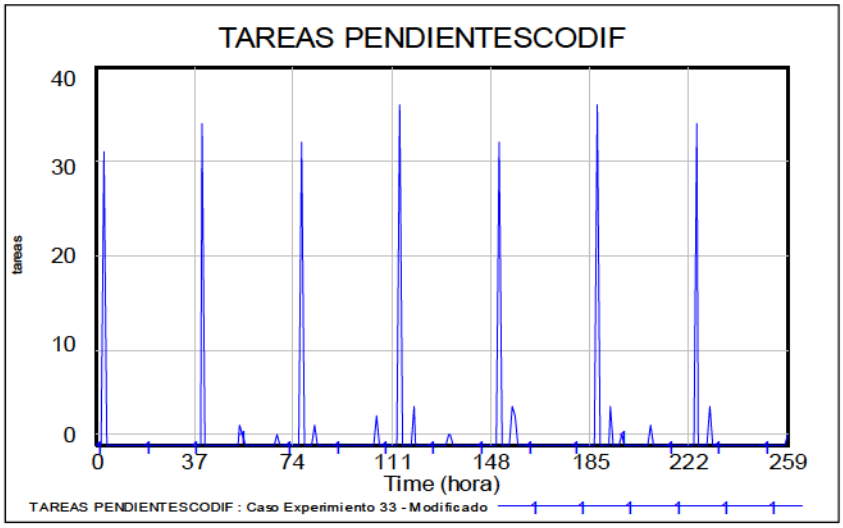


Figura 142 – Experimento 4 – Tareas Totales Pendientes de Codificación

Otra de las variables del modelo que incide en el desarrollo normal del proyecto y de la codificación es el nivel de ausentismo de los integrantes. A continuación en la Figura 143 se presentan las inasistencias no acordadas que se dieron a lo largo del proyecto.

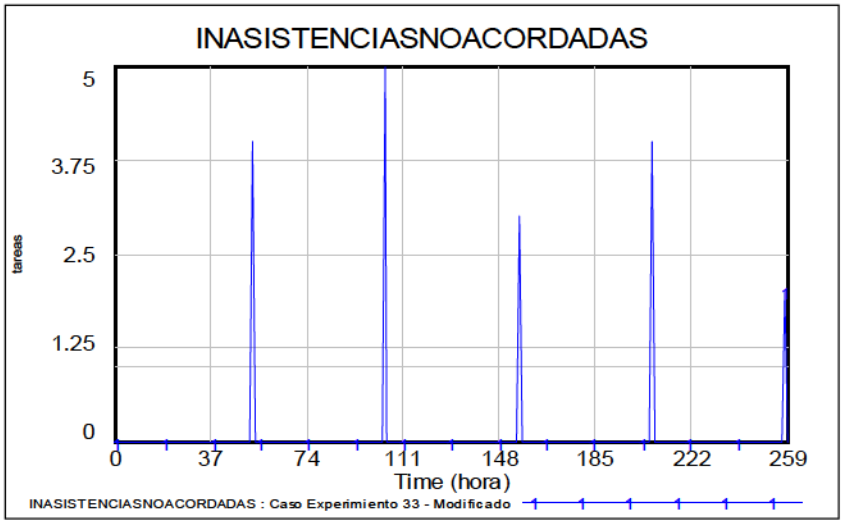


Figura 143 – Experimento 4 – Inasistencias no Acordadas

Con origen en la cantidad de Tareas Planificadas más las tareas que se suman a esta, surgen las tareas que deben ser codificadas. En la Figura 144 (ref. 2) se observa como las diferentes tareas se van codificando a medida que el tiempo avanza.

Se resalta en esta figura que al momento en que aparecen las tareas a Recodificar, producto de los errores el número de tareas que deben ser codificadas aumenta lógicamente haciendo que se generen pequeños picos en el gráfico.

Dada esta situación y manteniendo la velocidad inicial estimada, es evidente que el número inicial de tareas a codificar no se completa al momento de finalizar el Sprint.

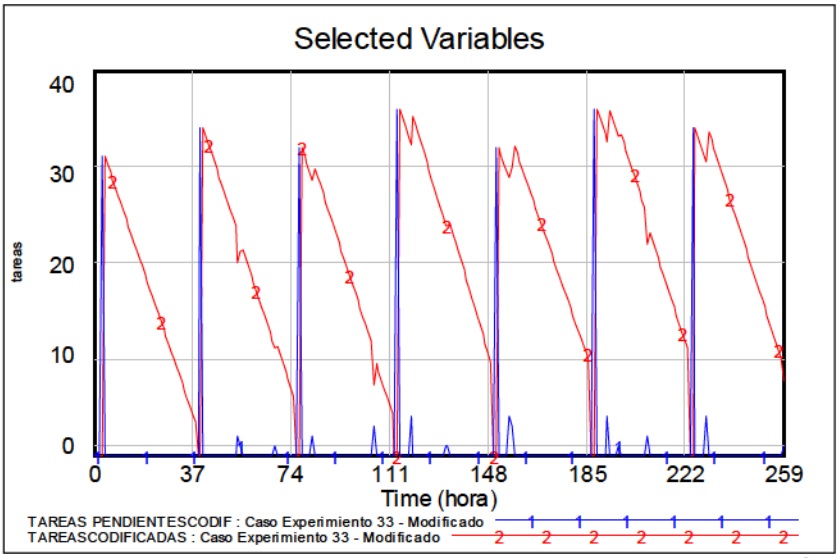


Figura 144 – Experimento 4 – Tareas Codificadas y Pendientes de Codificación

6.2.5.4 Políticas Propuestas

Dadas las situaciones que se presentaron durante las corridas bajo las condiciones iniciales se presenta a continuación una posible solución al problema, y consiste en realizar una variación en la velocidad diaria de trabajo de aproximadamente el 3%, aumentando así la velocidad de trabajo en cada uno de los Sprints del proyecto.

En primer lugar se muestra de manera superpuesta las variables que representan las Tareas pendientes de codificación (ref. 2), las tareas codificadas (ref. 3) y el Burdownchart (ref. 1). Se observa en la figura que el incremento en la velocidad hace que la codificación de tareas finalice antes de lo previsto, dado que el Burdownchart fue estimado con una velocidad y las tareas se desarrollaron en base a otra velocidad.

Se observa además en la Figura 145 que con el ajuste mencionado todas las tareas se codificarían al final de cada Sprint.

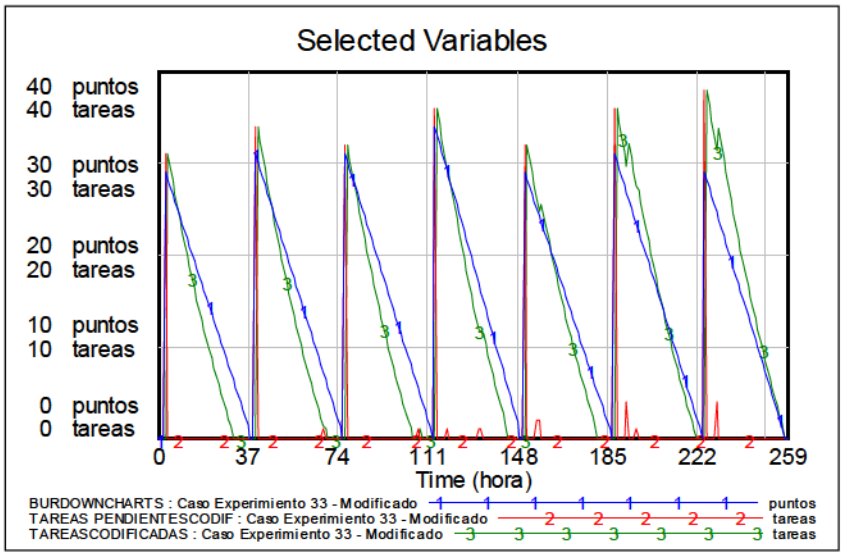


Figura 145 – Experimento 4 – Mejora – Tareas Codificadas y BurdownCharts

De manera similar y como consecuencia de las tareas codificadas en la Figura 146 se observa como la cantidad de Pruebas a Diseñar coincide con las Tareas Codificadas,

respetando además que el número de Pruebas a Diseñar se finalizan antes del comienzo del siguiente Sprint.

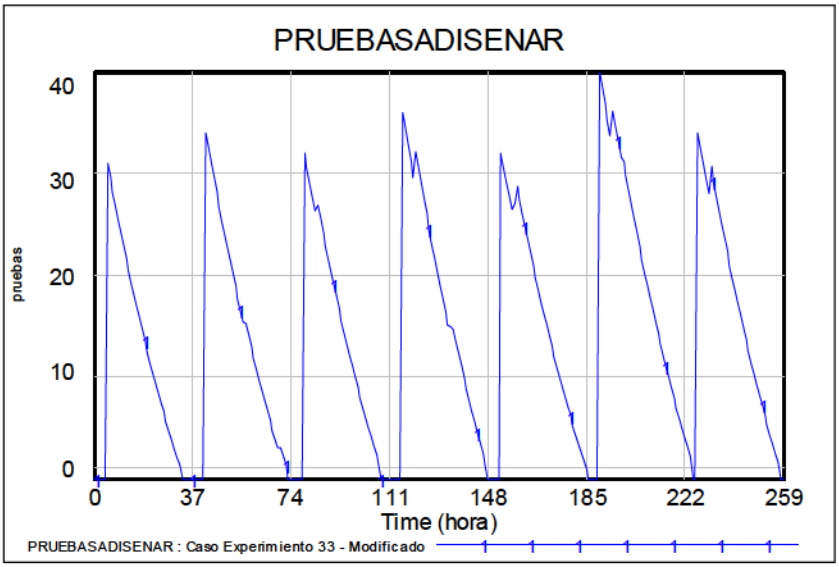


Figura 146 – Experimento 4 – Mejora – Pruebas a Diseñar

Dado que el número de pruebas diseñadas para cada Sprint logra completarse, es posible desarrollar las diferentes pruebas de Codificación a las tareas. En la Figura 147 se presenta el resultado para las pruebas realizadas. Superpuestos aparecen los resultados de las pruebas donde se detectaron errores en las tareas y aquellas tareas que no presentaron errores. A diferencia de otros casos es más difícil observar la incidencia de los errores en el gráfico de tareas sin errores, pero se denotan igualmente algunas caídas.

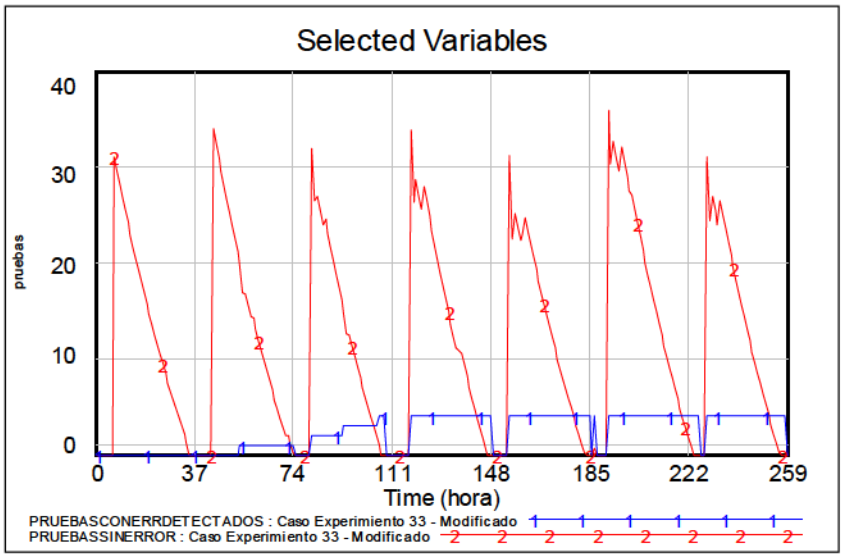


Figura 147 – Experimento 4 – Mejora –Pruebas Con Errores y Sin Errores de Codificación

Como anteriormente se observó en los Diagramas de Forrester [19] con las tareas que no presentaron errores por Codificación se procede a la generación de las pruebas de integración con las mismas.

Dado que para el presente escenario se consideró una tasa de error del 15% dicha tasa incide en las tareas produciendo el resultado de la Figura 148. Se observa en esta figura que existe una diferencia entre las pruebas de integración diseñadas y la cantidad

de pruebas sin errores, pero también se observan las tareas que si presentaron errores de integración y se corresponden con dicha diferencia.

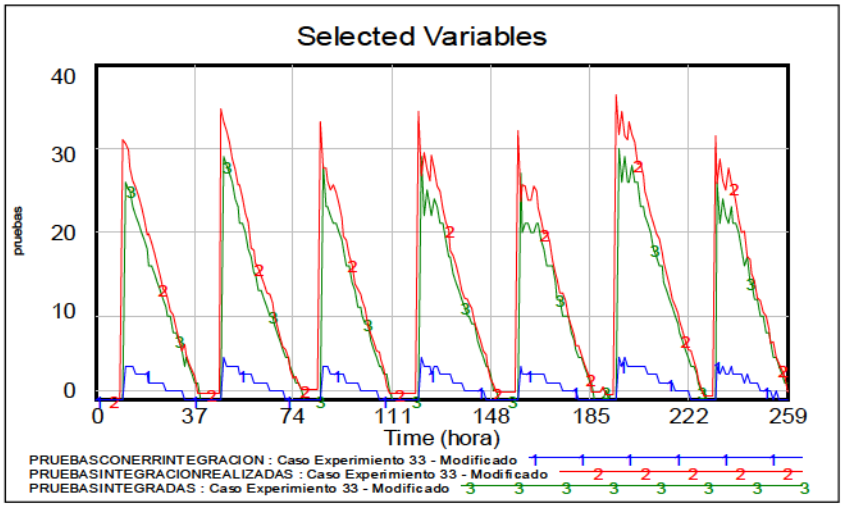


Figura 148 – Experimento 4 – Mejora – Integración de Tareas

En este experimento como se mencionó al inicio, se tienen en cuenta las variables relacionadas con los integrantes del Team más precisamente al número de integrantes y la experiencia del Team.

Dado que el Team comenzó con 6 integrantes, de los cuales 4 son Juniors Promocionales y 2 Juniors Contratados y el número pretendido era de 8 se decidió realizar la búsqueda de 2 candidatos Juniors Contratados que completen el número ideal.

En la Figura 149 se observa el estado inicial del Team. Luego de efectuada la búsqueda y completado el número ideal de Integrantes, el comportamiento que presentaron las variables de los integrantes se observa en la Figura 150. En esta última figura se observa que el número de integrantes Juniors Contratados pasa de 2 a 4, igualando la cantidad de Juniors Promocionales.

Además como únicamente se decidió integrar Juniors al Team, a la variable porcentajeSeniorsBuscados se le asignó el valor 0(cero) indicando así que no se requieren de Seniors.

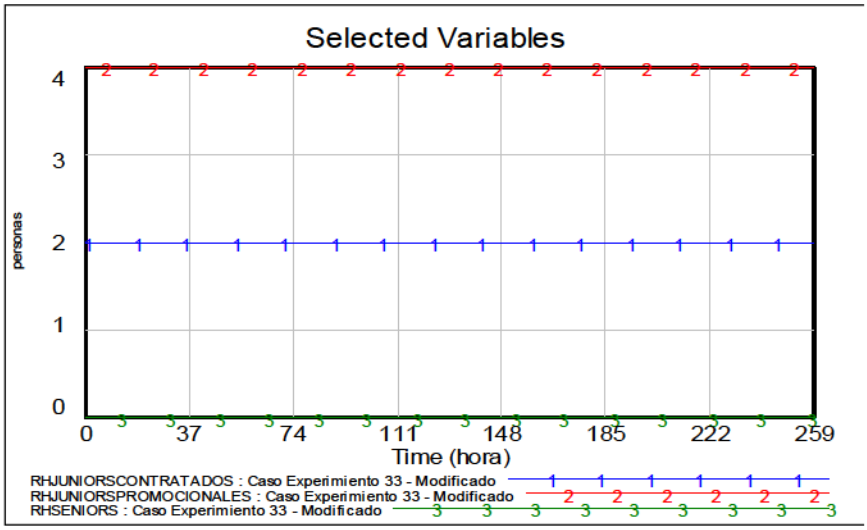


Figura 149 - Experimento 4 - Team Inicial

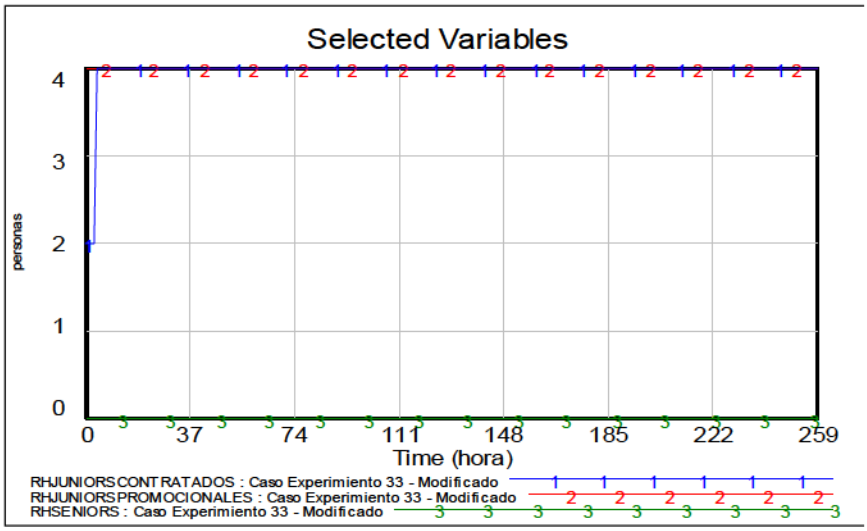


Figura 150 – Experimento 4 – Team Modificado

En las Figura 151 y Figura 152 se aprecian las curvas de experiencia del Team para el proyecto simulado. En la Figura 151 se ve la experiencia del Team donde no se producen modificaciones en el número de integrantes, considerando la situación inicial.

En la Figura 152 se observa como la experiencia en un determinado momento sufre una caída y luego vuelve a crecer levemente. Dicha caída se produce en el momento en el que el Team está adquiriendo experiencia y los nuevos integrantes Juniors contratados pasan a formar parte del Team para completar el número ideal de Integrantes. Luego de esto el Team continúa con el proceso normal de adquisición de experiencia.

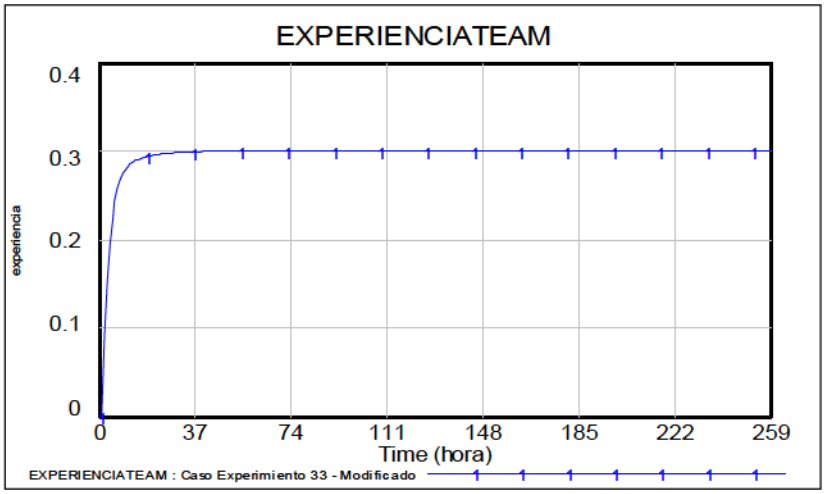


Figura 151 – Experimento 4 – Experiencia Team Inicial

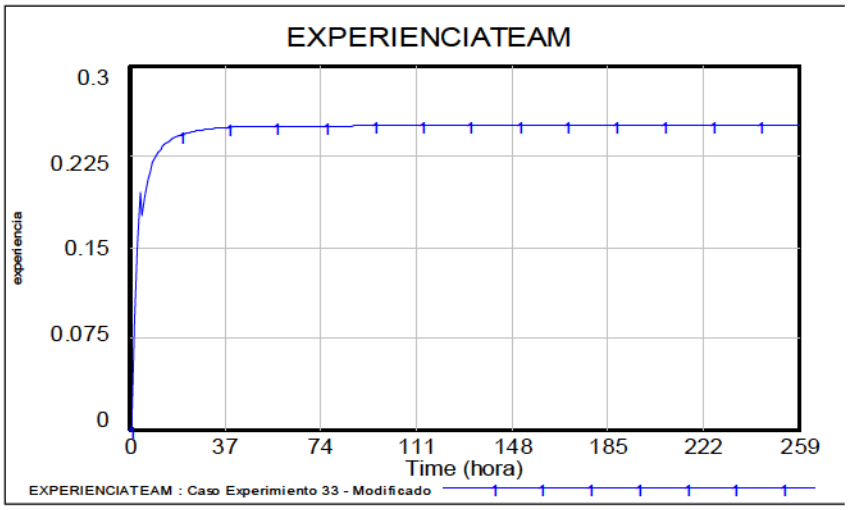


Figura 152 – Experimento 4 – Experiencia Team Modificado

A modo de complementar las figuras anteriores, se presenta en la Figura 153 de manera superpuestas el gráfico de experiencia del Team y el gráfico que muestra el momento en el que los nuevos integrantes Juniors Contratados pasan a formar parte del Team. Se observa que en el momento justo en el cual se produce la entrada de los integrantes la experiencia del Team sufre una baja.

Esta situación se relaciona con la Ley de Brooks presentada sección 2.1.5 la cuál menciona que el ingreso de nuevos miembros al Team puede generar un retraso en el proyecto en lugar contribuir al avance del mismo.

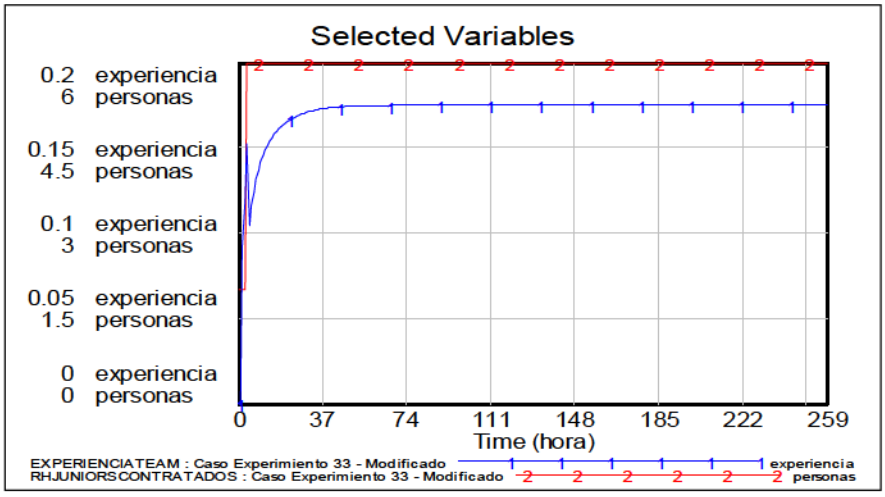


Figura 153 – Experimento 4 – Influencia Contratación y Experiencias Team

Otra de las variables del modelo y que influye en el cansancio del Team son las horas trabajadas a lo largo del proyecto. El comportamiento de la variable de nivel HORASTOTALESTRABAJAS se observa en la Figura 154.

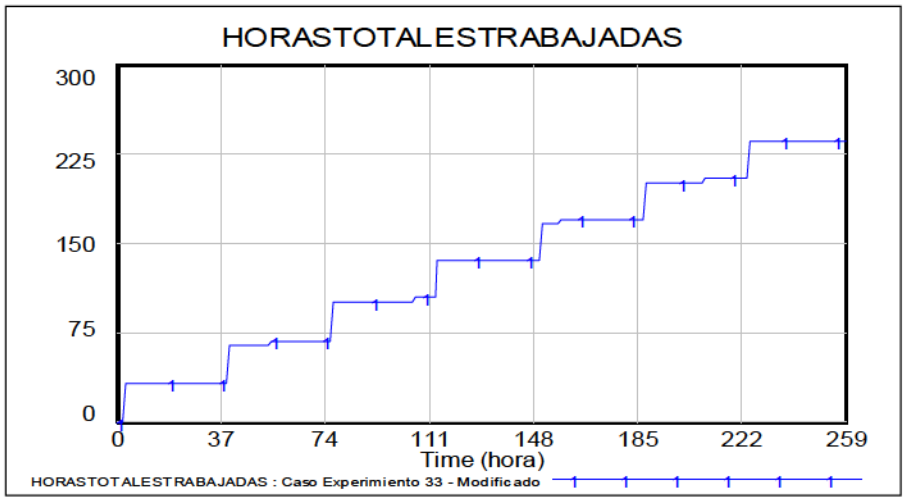


Figura 154 – Experimento 4 – Horas Totales Trabajadas por el Team

6.2.5.5 Conclusión

El presente experimento presentó dentro de las características principales una duración inicial de 245 horas, un número elevado de errores y haber comenzado con un número de integrantes diferente al ideal. En función de estas características, las mejoras propuestas pueden ser en dos sentidos, uno relacionado al desarrollo de tareas y el otro vinculado a la composición del Team.

Frente a las situaciones planteadas inicialmente en el experimento el Scrum Master podría utilizar el modelo en los Sprint Review o en los Scrum Daily a fin de evaluar mejoras como las planteadas o bien ir haciendo ajustes diariamente sobre la velocidad de trabajo. Por otro lado viendo lo sucedido frente a la incorporación de los nuevos integrantes Juniors Contratados y de la Ley de Brooks (sección 2.1.5) podría decidir también no realizar la incorporación de los integrantes Juniors y mantener el número inicial de integrantes.

6.3 Análisis de Sensibilidad

El objetivo de esta sección es ver el comportamiento del modelo frente a pequeños cambios en los valores de alguno de parámetros y variables del modelo y hasta donde es posible alterarlos sin comprometer la estabilidad del mismo.

Teniendo en cuenta que una de las principales variables es la Experiencia del Team, se presentan dos casos dentro de los que se encuentra encuentran: Tasa de Errores de Integración y Abandono del Team. A su vez estos casos presentan una serie de valores combinados que generan diferentes escenarios que permiten ver como en esas situaciones particulares se comportarían las variables involucradas y como responderían ante dichos cambios. En esta sección se presentan dos casos de análisis de sensibilidad. Los parámetros y condiciones iniciales de los experimentos están basados en el caso de Experimentación presentado en la sección 6.2.4.

Hay que aclarar que al estar utilizando la Versión Académica, Vensim PLE, los experimentos y análisis que se presentan en esta sección fueron realizados de forma manual ya que se desconocía al momento de la presentación del tema de la tesis que la versión académica no poseía esta funcionalidad. La Versión licenciada de Vensim si posee una herramienta para realizar este tipo de análisis de forma automatizada.

6.3.1 Análisis Sensibilidad 1

El primero de los análisis que se presenta involucra los Subsistema de Promoción de Recursos Humanos, el de Producción y de Adquisición de Experiencia. El análisis que se desarrolla tiene como eje central la experiencia del Team, y de cómo esta influye en la tasa de errores que se producen al momento de la codificación de tareas, dado que el Team tiene cierta capacidad para poder sobrellevar la presión que sufren sus integrantes en la medida que se acerca la finalización del Proyecto.

Se presentan aquí 4 opciones en la composición de los integrantes del Team:

Opción 1: Todos los integrantes del Team son Seniors. La experiencia del Team crece rápidamente alcanzando un valor máximo de 0.9 puntos de experiencia.

Opción 2: Conformado por 6 Integrantes Juniors Promocionales. Se nota un crecimiento similar al del Caso 1, aunque la diferencia del crecimiento es más lenta al inicio del proyecto, para luego situarse en un valor similar.

Opción 3: Conformado por 6 integrantes Juniors Contratados. Presenta una curva de crecimiento más lenta que las anteriores variantes, pero además, es muy baja en cuanto al valor que toma rondando 0.1 de experiencia.

Opción 4: conformado por 2 integrantes de cada nivel de experiencia. Se logra un nivel medio de experiencia lo que denota que la experiencia total del Team es media y cercana a 0.6 puntos de experiencia.

El resultado de correr el modelo con cada una de estas opciones se observa en la Figura 155.

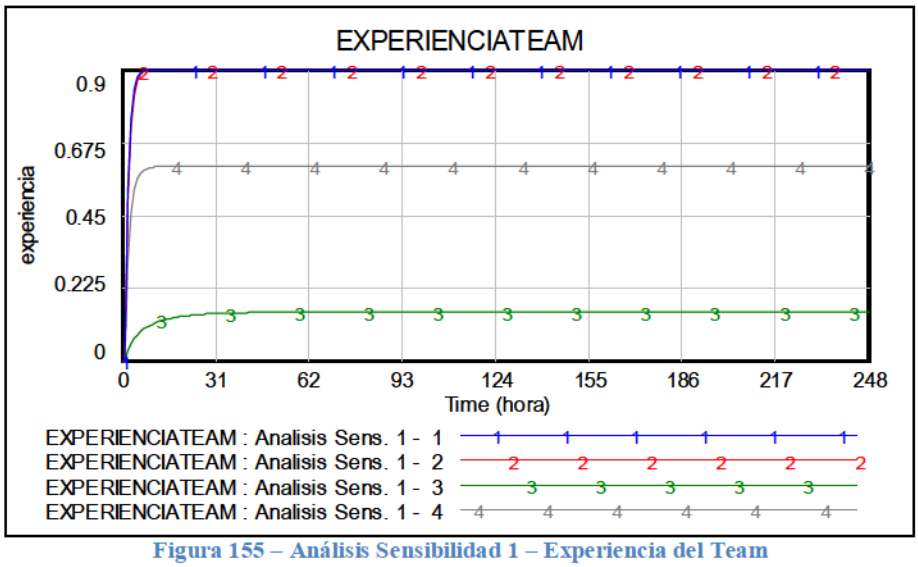


Figura 155 – Análisis Sensibilidad 1 – Experiencia del Team

Dentro de las variables del modelo tasaErrPorPresionPlazo es una de las que directamente sufre la influencia de la Experiencia del Team. Está representa la tasa de errores en función de la presión que sufre el Team, y de su valor dependerá la cantidad de errores que se produzcan al momento de codificar las tareas.

A continuación se presentan los resultados y los diferentes comportamientos que presentó la Tasa de Error, a partir de la Presión en el Plazo y de la Experiencia de los Integrantes del Team. El comportamiento de la variable tasaErrPorPresionPlazo se observa en la Figura 156.

En el caso 1 donde todos son Seniors, el momento en el cual las tareas con error se presentan están mucho más cerca del instante en el que finalizará el proyecto.

En el caso 2, se nota un crecimiento similar al del Caso 1, aunque el momento de aparición de los errores es apenas anterior al momento en el que ocurren en el caso 1.

El caso 3 donde todos son Juniors contratados, el momento en el cual las tareas con errores se presentan está mucho más cerca del instante en el que se inicia el proyecto.

En el caso 4 donde hay 2 integrantes de cada nivel de experiencia, se observa que el momento en el cual las tareas con error se presentan está un punto medio del proyecto.

Las consecuencias de la aparición más temprana o más tardía de la probabilidad de errores, hace que la cantidad de tareas con errores sea mayor o menor respectivamente.

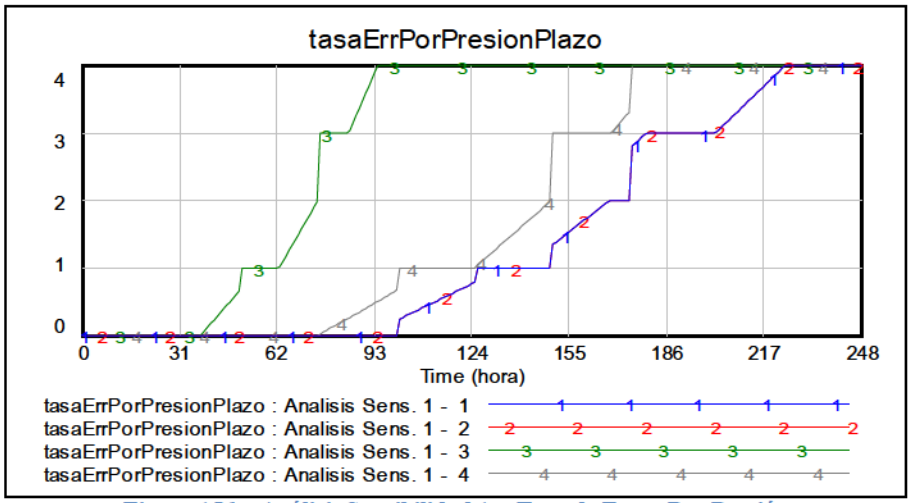


Figura 156 – Análisis Sensibilidad 1 - Tasa de Error Por Presión

En la siguiente Figura 157 se presenta a modo de ejemplo y para una mejor lectura del comportamiento, el caso 1 (ref. 1) y el caso 1 (ref. 3) del análisis de sensibilidad se presenta en la Figura 158. Se observa de manera superpuesta el gráfico de la tasa de errores por presión que sufre el Team y la cantidad de tareas con error que ocurrieron en ese momento.

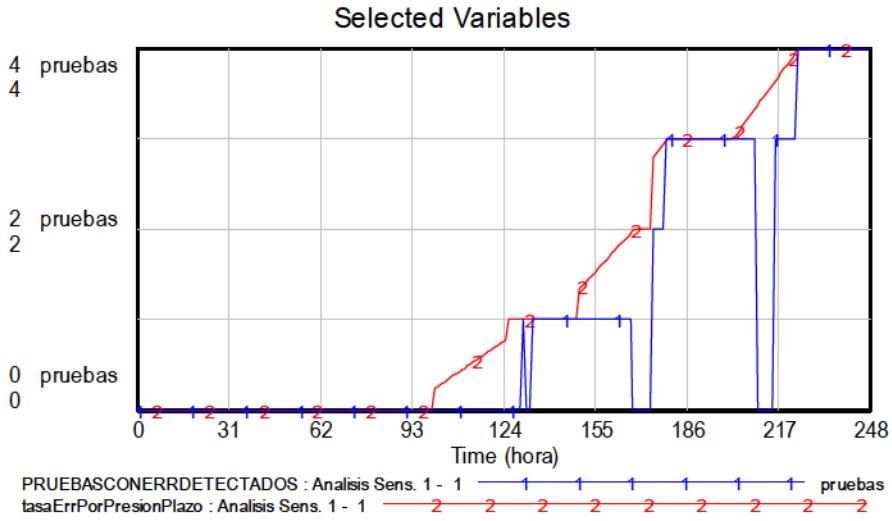


Figura 157 - Análisis Sensibilidad 1 - Caso 1 - Tasa de Error

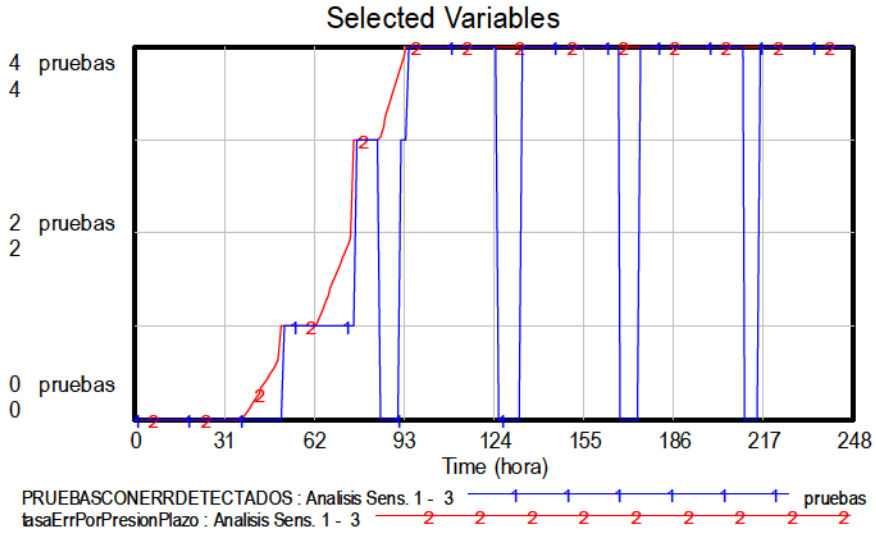


Figura 158 - Análisis Sensibilidad 1 - Caso 3 -Tareas con Errores

6.3.2 Análisis Sensibilidad 2

El presente caso de análisis de sensibilidad muestra como las variaciones en número y experiencia personal de los integrantes, influye en la experiencia total del Team. Es por ello que se realizaron 3 opciones con diferentes variantes, donde básicamente se modificó el número y la composición de los integrantes en diversos momentos del

proyecto, con el objetivo de ver cuán sensibles es la experiencia total del Team a estas modificaciones y hasta donde el modelo sigue siendo aceptable.

6.3.2.1 Opción 1

En este caso el número de integrantes ideales para iniciar el proyecto es de 8, pero se inicia con solo 5 integrantes Seniors. Las variantes aquí propuestas consisten en la búsqueda de los 3 integrantes restantes hasta completar el número de integrantes requeridos.

Las variables y los parámetros que permiten iniciar este caso de análisis se enumeran junto a sus valores:

auxMaximoCandidatos: 15.
porcentajeSeniorsBuscados: 1 (100%) –Primera Opción-
porcentajeSeniorsBuscados: 0.5 (50%) –Segunda Variante-
porcentajeSeniorsBuscados: 0.5 (50%) –Tercera Opción-
porcentajeSeniorsBuscados: 0 (0%) –Cuarta Opción-

En la Figura 159 se muestra la variación del número de integrantes Juniors. Se observa como la cantidad de integrantes de este nivel va aumentando en la medida que los Juniors contratados completan el número requerido de integrantes del Team.

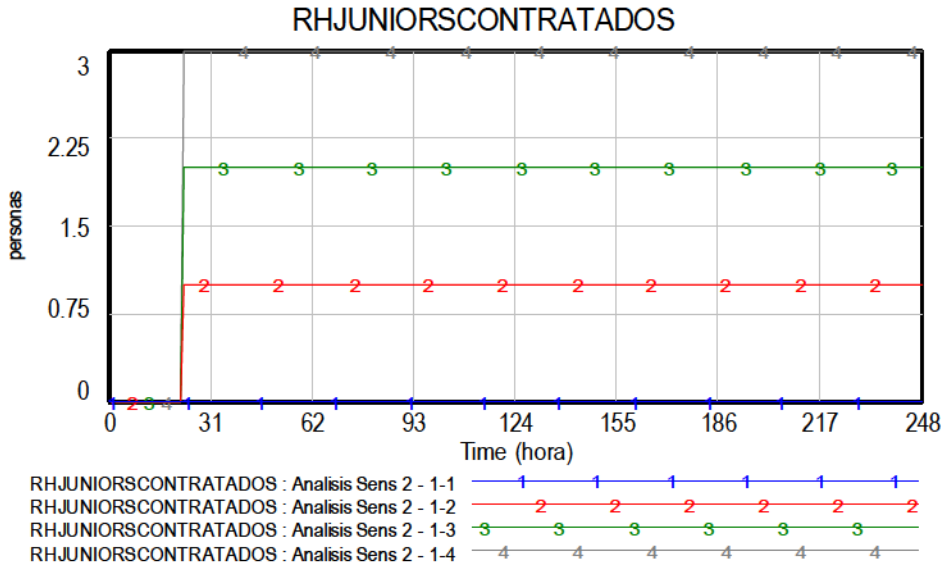


Figura 159 – Análisis Sensibilidad 2 Caso 1 – Rec.Humanos Juniors Contratados

En la Figura 160 se observa como la cantidad de integrantes Seniors fue descendiendo de 8 a 6. Dado que este nivel empezó con 5 integrantes, en las diversas situaciones se fueron incorporando cada vez menos integrantes Experimentados, dado que el porcentaje de búsqueda se fue reduciendo.

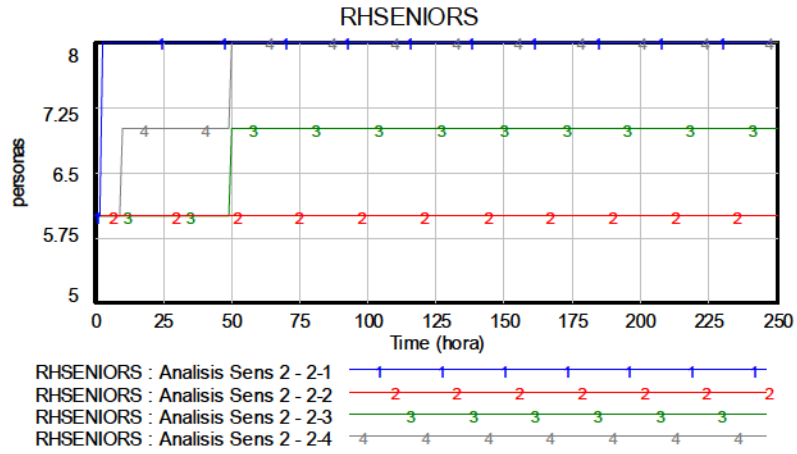


Figura 160 – Análisis Sensibilidad 2 Caso 1 – Rec.Humanos Seniors

En función del comportamiento de las variables de nivel de los integrantes Seniors y JuniorsContratados la variable que representa la experiencia del Team tiene un comportamiento acorde.

En la Figura 161 se puede ver como la experiencia del Team desciende en la medida que el número de integrantes Juniors aumenta, y disminuye la cantidad de Seniors presentándose la idea expuesta en la sección 1.6 donde se presentó la ley de Brooks.

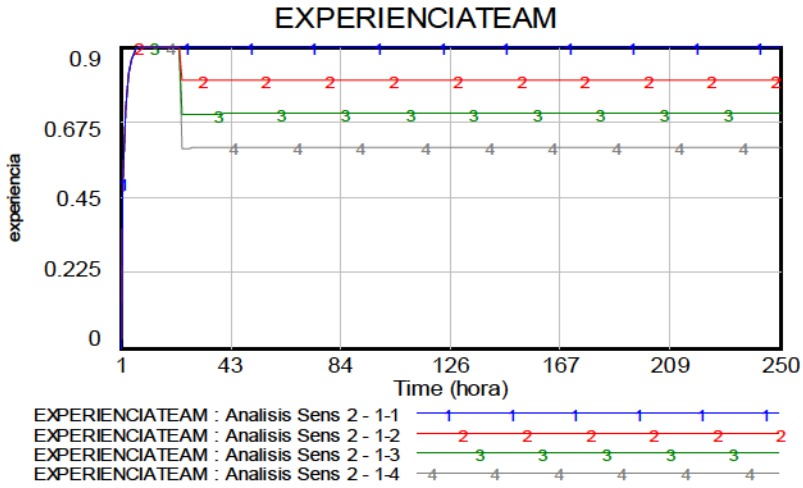


Figura 161 – Análisis Sensibilidad 2 Caso 1 – Experiencia Team

6.3.2.2 Opción 2

Para este caso el número de integrantes ideales elegido es de 8. El proyecto se inicia con 6 Seniors y 2 Juniors Contratados por lo que el Team está completo.

En las opciones de este caso, se realizaran diferentes promociones tanto de Juniors Contratados a Juniors Promocionales, como de estos últimos a Seniors. Los espacios de promociones estarán separados por un plazo de 40 horas.

Para este Caso 2 los valores que toman los parámetros que permiten iniciar el análisis se presentan junto a sus variables:

auxMaximoCandidatos: 15
porcentajeSeniorsBuscados: 1 (100%)
esperaEnPromocion: 1 (8 horas)
tasaAvanceASeniors: 0 (0%)

Primera Opción:
tasaAJuniors: 0. 5 (50%)
tasaASeniors: 0 (0%)
Segunda Opción:
tasaAJuniors: 1 (100%)
tasaASeniors: 0 (0%)
Tercera Opción:
tasaAJuniors: 1 (100%)
tasaASeniors: 0.5 (50%)
Cuarta Opción:
tasaAJuniors: 1 (100%)
tasaASeniors: 1 (100%)
demoraPaseAJuniors: 40 horas

En la a Figura 162 se pueden observar las variaciones del número de integrantes Juniors Promocionales para los diversos casos. Es válido recordar que las opciones se fueron haciendo sobre el mismo número de integrantes ingresados. Por lo tanto en la figura se observan picos que representan cada uno de los integrantes que pasaron al siguiente nivel.

Otra característica que se observa en la figura es que a diferencia de lo explicado recientemente, el comportamiento de la variable es lineal. Esto se debe a que en ese momento no se produjeron pases al siguiente nivel lo que indica que el número permaneció estable a lo largo de la corrida.

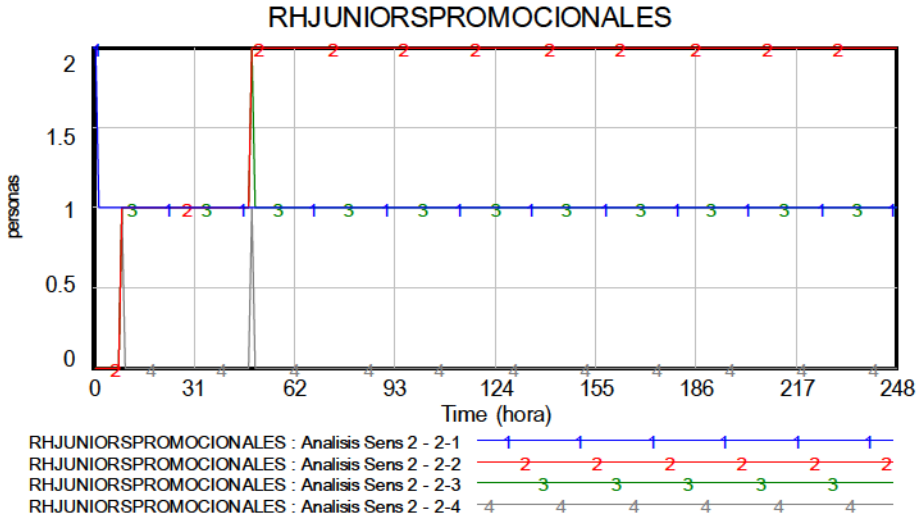


Figura 162 – Análisis Sensibilidad 2 Caso 2 – Rec. Humanos Juniors Promocionales

La Figura 163 muestra el comportamiento de la variable que representa los integrantes Seniors del Team. A diferencia del comportamiento visto en la variable de Juniors (Figura 164), aquí al no producirse pases a otro nivel de experiencia el comportamiento si bien es escalonado dado que ingresaron nuevos integrantes al nivel, una vez que esto ocurre el comportamiento es lineal representando la estabilidad de la cantidad de integrantes.

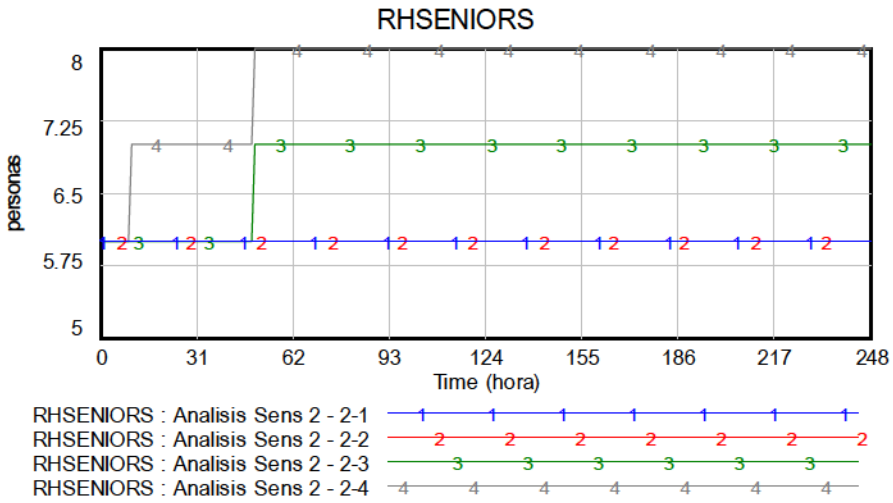


Figura 163 – Análisis Sensibilidad 2 Caso 2 – Rec. Humanos Seniors

Como se mencionó en el Subsistema de Adquisición de Experiencia (sección 5.11), este depende del número de integrantes existentes en cada nivel. Se observa en la Figura 164 como en la medida que aumenta el número de integrantes Seniors debido a las promociones de los integrantes JuniorsPromocionales la variable que representa la experiencia total del Team aumenta.

Los escalones que se observan en algunos de los casos, indica el momento en el cuál ocurrieron las promociones.

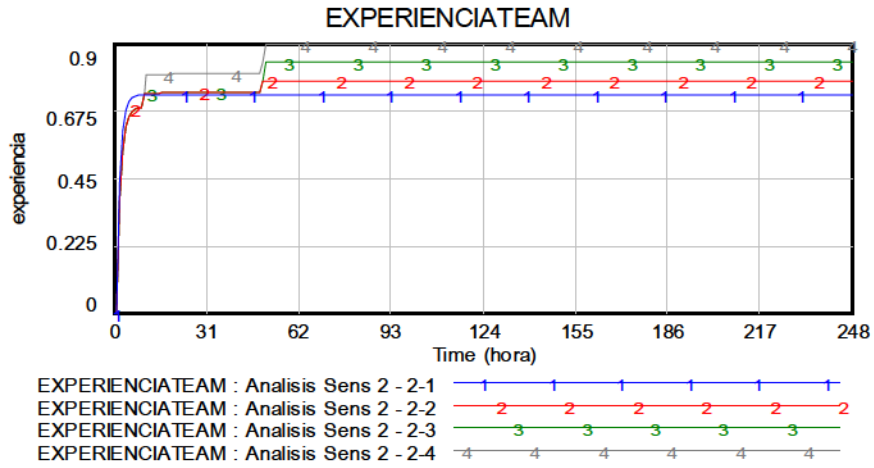


Figura 164 – Análisis Sensibilidad 2 Caso 2 – Experiencia Team

6.3.2.3 Opción 3

En este caso el número de integrantes ideal es de 10 personas y el Team está compuesto por 6 Seniors, 2 JuniorsContrados y 2 Juniors Promocionales. Se propusieron opciones, donde en cada una se muestra el efecto en la experiencia del Team tras el abandono de los integrantes. Se destaca que no se produce la reincorporación de aquellos integrantes Juniors Contratados que abandonan el Team. Y también es relevante mencionar que los casos son acumulativos, por lo tanto la salida de un integrante y salvo caso que se considere la búsqueda de un reemplazo, el Team permanecerá con un integrante menos.

Para el Caso 3 los parámetros que permiten iniciar el análisis enumeran junto a sus valores.

auxMaximoCandidatos: 15

porcentajeSeniorsBuscados: 1(100%)
esperaEnPromocion: 1 (8 horas)
tasaAvanceASeniors: 0 (0%)
tasaAJuniors: 0 (0%)
demoraPaseAJuniors: 0

Las opciones que se muestran en la Figura 9-11 y que se fueron realizando de manera acumulativa en cuanto a la salida y recontratación de los integrantes, se realizaron en función de los siguientes cambios en la conformación del Team:

1- 0.5 (50%) abandono Juniors contratados a las 40 Horas.

2- 0.5 (50%) abandono Juniors promocionales a las 72 Horas.

3- 0.2 (20%) abandono Seniors a las 96 Horas.

4- 0.5 (50%) abandono noveles a las, 0.5 (50%) abandono promocionales a las 104 Horas.

5- 1 abandono Experto a las 112 Horas.

La Figura 165 muestra como a lo largo de las diferentes opciones del presente caso de análisis de sensibilidad la variable RHJUNIORSCONTRATADOS toma valores acordes a lo planificado, produciéndose escalones descendentes en los casos donde los integrantes abandonaron el Team.

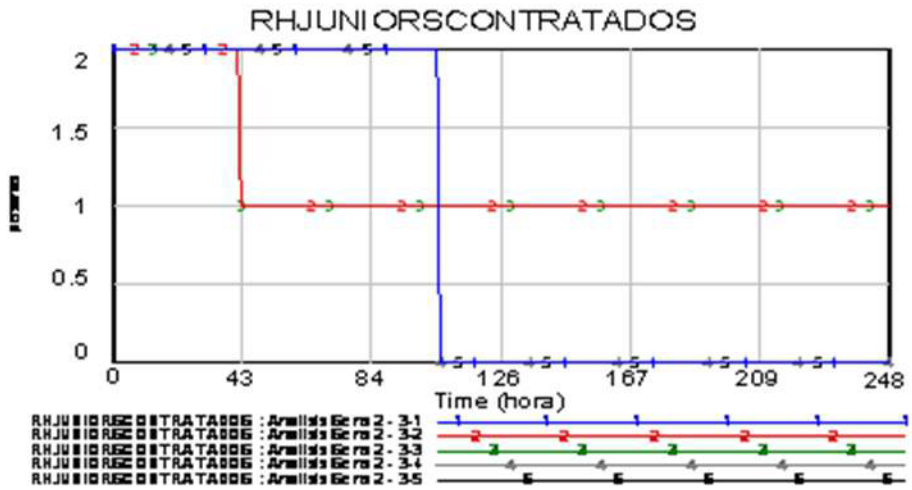


Figura 165 – Análisis Sensibilidad 2 Caso 3 – Rec. Humanos Juniors Contratados

La Figura 166 muestra como a lo largo de las diferentes variantes del presente caso de análisis de sensibilidad la variable RHJUNIORSPROMOCIONALES toma valores acordes a lo planificado, produciéndose escalones descendentes en los casos donde los integrantes abandonaron el Team.

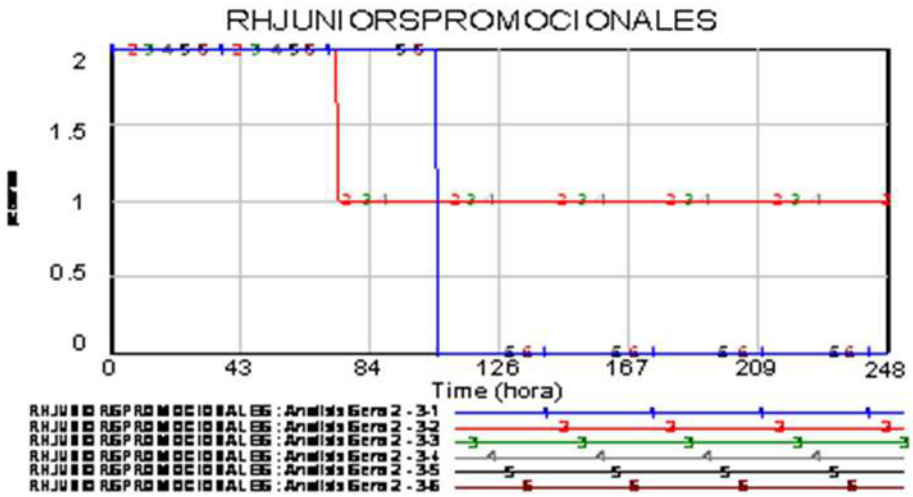


Figura 166 – Análisis Sensibilidad 2 Caso 3 – Rec. Humanos Juniors Promocionales

La Figura 167 muestra como a lo largo de las diferentes variantes del presente caso de análisis de sensibilidad la variable RHSENIORS toma valores acordes a lo planificado. Se observa en la Figura que el comportamiento de la variable presenta caídas (se produce el abandono), mesetas (momento de la búsqueda del reemplazante), y luego ascensos (ingreso del nuevo Experto al Team).

El momento de meseta y ascenso más marcado se da en la referencia 4, que es el momento en el cuál se tardaron 40 Horas en encontrar los reemplazos para el Team llegando nuevamente al número ideal de 10 personas.

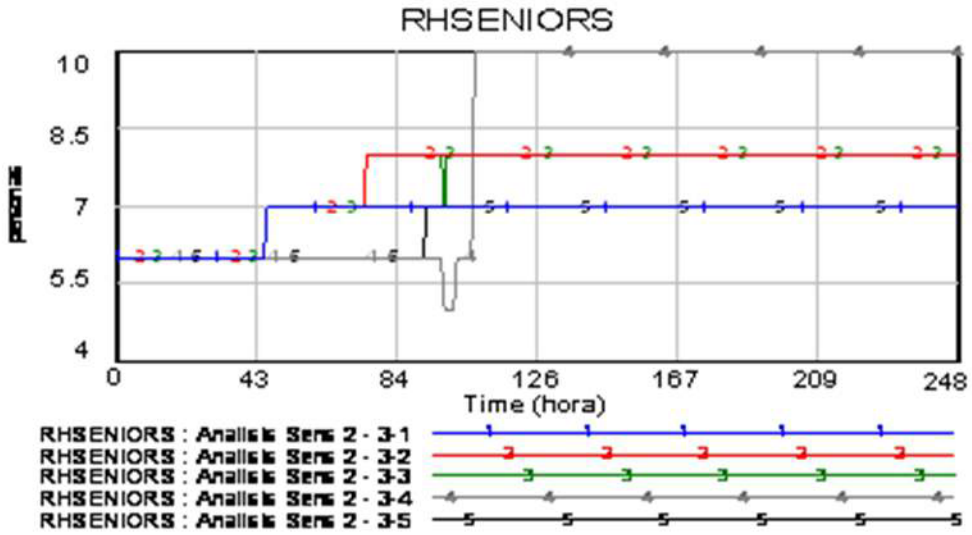


Figura 167 – Análisis Sensibilidad 2 Caso 3 – Rec. Humanos Seniors

La Figura 168 muestra el comportamiento de la variable de la Experiencia del Team, y como a lo largo de las diferentes variantes propuestas para el presente caso de análisis de sensibilidad la variable toma valores acordes al número de integrantes y su nivel de experiencia.

En la variante 4 se presenta una situación particular donde luego de un pequeño descenso de la experiencia esta sube hasta el máximo, esto se da en el momento en el que el Team se conforma con la totalidad de sus integrantes y donde la mayoría son Seniors.

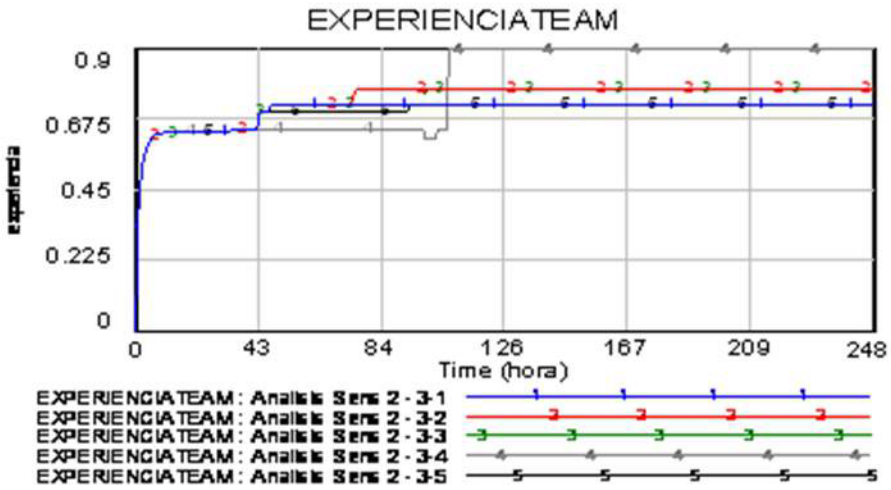


Figura 168 – Análisis Sensibilidad 2 Caso 3 – Experiencia Team

6.3.3 Conclusión del Análisis de sensibilidad

Los diferentes casos presentados en el análisis de sensibilidad expuesto pone de manifiesto la relación existente entre la experiencia del Team y la conformación de los diferentes niveles de Experiencia modelados. Esta relación entre la cantidad, nivel de experiencia y experiencia total del Team se observa en los resultados obtenidos en los diferentes experimentos. Este análisis de sensibilidad respalda los resultados obtenidos en los diferentes Casos de Experimentación realizados en la sección 6.2 y permite ver de qué manera influyen las variables que relacionadas con la experiencia del Team y los parámetros que especifican los diferentes niveles de experiencia de los integrantes del Team.

Conclusiones

En este trabajo se diseñó y construyó un Modelo Dinámico de Simulación para Proyectos de Software que se gestiona siguiendo la metodología Scrum. Este modelo puede ser de utilidad para los Scrum Master y el Team puedan analizar el efecto del uso conjunto de la metodología Scrum, Bloques de Tiempo, Artefactos y Reglas en la gestión de los mismos en diferentes escenarios. La flexibilidad de modelo permite modificar los valores de los parámetros tanto al inicio de la simulación como al momento de la ejecución de la misma. Dentro de los parámetros que se pueden establecer previos al inicio de cada simulación se encuentran: la duración y la velocidad de cada Sprint, la velocidad estimada de desarrollo de las tareas, Factores de Cansancio, de Presión en el plazo, Cantidad de integrantes del Team según su experiencia en la metodología y las tareas extras que se prevén puedan surgir. A través de la modificación de valores de los parámetros el usuario puede establecer o modificar la cantidad de integrantes del Team que abandonan el proyecto, clasificar al Team mediante la asociación de estos a en base a su experiencia en Scrum como Juniors o Expertos, cambiar la cantidad de horas estimadas de duración del proyecto, generar horas extras e inasistencia de los integrantes de manera determinística o pseudoaleatoria, entre otros.

En cuanto a la metodología de dinámica de sistemas utilizada para construir el modelo, se ha validado su aplicación en este tipo de simulaciones.

La validación permitió ajustar el modelo para que su comportamiento pueda reproducir con efectividad el comportamiento de los casos reales estudiados. Los casos de experimentación dieron la posibilidad de probar políticas alternativas para resolver situaciones que se puedan presentar en proyectos gestionados con Scrum.

Finalmente, luego de validar y experimentar con el modelo, se ha llegado a la conclusión de que el mismo puede ser utilizado como herramienta para evaluar el impacto de políticas alternativas de gestión y la detección de cuellos de botella en el desarrollo de proyectos gestionados con la metodología Scrum.

En comparación a los trabajos relacionados, este se destaca por su especificidad relacionada con los desarrollos de proyectos de software que utilizan Scrum, donde se han modelado características propias y únicas de esta metodología.

Como conclusión final se puede decir que el modelo cumple con su objetivo de ser de utilidad como una herramienta para el Scrum Master y el Team a la hora de analizar el efecto del uso conjunto de la metodología Scrum en proyectos de desarrollo de software. De la misma manera, puede servir en cada una de las reuniones de planificación y para discutir con el Proyect Owner de manera objetiva sobre el avance del proyecto y los riesgos e implicancias de cada una de las decisiones.

Lo que diferencia a este trabajo de otros relacionados es que se han modelado las características esenciales de la metodología Scrum aplicada a proyectos de desarrollo de software.

En este sentido el modelo desarrollado puede ser de utilidad para entrenar a futuros Scrum Másters sobre las decisiones de gestión que puedan tomar en cualquier situación que se les presente.

En cuanto a la versión del software utilizado para construir el modelo podemos decir que la Versión de Vemsim PLE utilizada es suficiente para modelar este tipo de problemas. Sin embargo esta versión de la herramienta Vensim solo tiene disponible la

posibilidad de realizar análisis de sensibilidad de forma automática en la versión comercial. Es por ello que este trabajo se realizó un análisis de sensibilidad manual.

Trabajos Futuros

Como trabajos futuros se planea Adicionar otros subsistemas que permitan ampliar el ámbito del modelo, y pueda ser utilizado como una herramienta que facilite aún más las tareas previas al inicio del proyecto al Scrum Master y al Team. Dentro de estos sub-sistemas podrían nombrarse: cálculo de costos, comunicación en el Team, Intercambio de Roles, Adquisición de Experiencia, por nombrar solamente algunos. Así también, se pretende avanzar con la construcción de modelos similares para otras metodologías consideradas ágiles como Cristal Clear y Crystal Orange [30] [29] o incluir prácticas como por ejemplo Test DrivenDevelopment [47].

Por otra parte se planea adaptar este simulador para poder utilizarlo en cátedras relacionadas con la Ingeniería de Software, con el fin de entrenar a futuros Scrum Masters y Teams.

Es también necesario construir bases de datos de Proyectos Scrum que contengan datos de proyectos de software reales llevados a cabo, ya que actualmente resulta difícil contar con datos post mortem de proyectos gestionados con métodos ágiles.

La utilización de una versión Licenciada de Vensim para realizar los análisis de sensibilidad de manera automatizada y poder importar registros de la base de datos de proyectos anteriores que sirvan de entrada al modelo.

Bibliografía

[1] Ken Schwaber and Jeff Sutherland, *Agile Software Development with Scrum*, 1st ed., 2001.

[2] Kim E. Van, Kishore Sengupta, and Luk N. Van., "Dynamics of Agile Software Development," 2009.

[3] Xiaoying Konga, Li Liu, and Chen Jing, "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment," *International Conference on Advances in Engineering 2011*, 2011.

[4] Tamara Kasiak and Diego Alberto Godoy, "Simulación de Proyectos de Software desarrollados con XP," *XIV Workshop de Investigadores en Ciencias de la Computación.*, 2012.

[5] Diego Alberto Godoy and Tamara Kasiak, Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP, Octubre 2012, XVIII Congreso Argentino de Ciencias de la Computación. Extensión: 10 p.

[6] Firas Glaiel, Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics, Junio 2012, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

[7] Bodje N'Kauh Nathan Regis, "Evaluation of the Most Used Agile Methods (XP, LEAN, SCRUM): With the Definition of Quality Developed by Toyota," *LAP Lambert Academic Publishing, , Germany*, 2012.

[8] Diego Alberto Godoy, Edgardo A. Belloni, Henry Kotynski, Hector H Dos Santos, and Eduardo Omar Sosa, "Simulando Proyectos de Desarrollo deSoftware Administrados con Scrum," in *XVI Workshop de Investigadores en Ciencias de la Computación RedUNCI*, Ushuaia, 2014.

[9] Diego Alberto Godoy, Henry Kotinski, Eduardo Omar Sosa, Edgardo Belloni, and Juan de Dios Benitez, "Evaluación de Alternativas de Gestión en Proyectos de Software Desarrollados con Scrum utilizando Dinámica de Sistemas"," in *XX Congreso Argentina de Ciencias de la Computación*, San Justo, Buenos Aires, 2014.

[10] Diego Alberto Godoy, Eduardo Omar Sosa, Edgardo Belloni, and Henry Kotinski, "Simulación Dinámica de Gestión de Tareas en Proyectos Desarrollados Con Scrum," in *II Congreso Nacional de ingenieria informatica/ingenieria de sistemas (CoNaIISI)*, Universidad Nacional de San Luis, San Luis, 2014, 2014.

[11] Javier Aracil, *Dinámica de Sistemas*, Primera Edición ed. Madrid, España, 1997.

[12] M. Minsky, *Matter, Minds and Models*, Primera ed., The MITT Press, Ed. Massachusetts, EEUU: The MITT Press, 1965.

[13] Thomas H Naylor, *Técnicas de Simulación en Computadoras*. New York, Estados Unidos: Limusa, 1977.

[14] J Barcelo, *Simulación de Sistemas Discretos*. España: I.D.E.F.E., 1996.

[15] R. Cao, *Introducción a la Simulación y a la Teoría de Colas*. La Coruña: Editorial Netbiblo, 2002.

[16] Wayne L. Winston, *Investigación de Operaciones. Aplicaciones y Algoritmos*.: Editorial Iberoamérica, 1.994., 1994.

[17] F. y Lieberman, G. Hillier, *Introducción a la Investigación de Operaciones*.: McGraw-Hill, 1997.

[18] Peter M. Senge, *La quinta disciplina en la práctica: cómo construir una organización inteligente*.: Grancia, Ediciones, S.A., 1995.

[19] Jay W. Forrester, *Industrial Dynamics - After the First Decade*, Massachusetts Institute of Technology Press, Ed. Michigan, Estados Unidos: Universidad de Michigan, 1961.

[20] VenSim. (2014, Jan.) http://vensim.com/. [Online]. http://vensim.com/vensim-personal-learning-edition/

[21] J Torrealdea, *Dinámica de Sistemas. Elementos y Estructuras de un Modelo.Construyendo modelos*.: Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco, 2003.

[22] Monica Talledo Jimenez, *Guía de los Fundamentos para la Dirección de Proyectos - (Guía del PMBOK)*, Cuarta Edición ed., PMI Publications, Ed. Newtown Square,

Pennsylvania, EE.UU.: PMI Publications, 2008.

[23] J. Gido and J. P. Clements, *Successful Project Management*, 5th ed., 2012.

[24] IMPA. (2014) International Proyect Management Assosiation. [Online]. http://ipma.ch/

[25] Ian Somerville, *Ingeniería de Software*, Septima ed., Pearson Educación S.A., Ed. Madrid, España: Pearson Addison Wesley, 2005.

[26] Jr Frederick P. Brooks, *The Mythical Man-Month Essays on Software Engineering*.: ADDISON-WESLEY, 1995.

[27] Kent Beck, *Una Explicación de la Programación Extrema. Aceptar el Cambio*,. ed., Addison Wesley, Ed. Madrid, España: Addison Wesley, 2002.

[28] Varios. (2014, Jan.) Agile Manifesto. [Online]. http://agilemanifesto.org

[29] Cockburn, Alistair, *Agile Software Development: The Cooperative Game*.: Addison-Wesley Professional, 2006.

[30] Alistair Cockburn, *Crystal Clear, A Human-Powered Methodology for Small Teams*.: Addison-Wesley Professional, 2004.

[31] J Highsmith and K Orr, *Adaptive Software Development: A Collaborative Approach to Managing Complex*.

[32] P Coad, E Lefebvre, and J De Luca , *Java Modeling In Color With UML: Enterprise Components and Process*.: Prentice Hall, 1999.

[33] J Stapleton, *Dsdm Dynamic Systems Development Method: The Method in Practice*.: Addison-Wesley, 1997.

[34] Hirotaka Takeuchi and Ikujiro Nonaka, New New Product Development Game. Harvard Business Review, Enero 01, 1986.

[35] Adriana Peralta, "Metodología SCRUM," Universidad ORT Uruguay, Montevideo, Uruguay, 2003.

[36] Abdel-Hamid T. and S. Madnick, *Software Project Dynamics: An Integrated Approach*.: Prentice-Hall software Series, 1991.

[37] Miguel Toro, Mercedes Ruiz Isabel Ramos, "Modelo Dinámico Reducido," Universidad de Sevilla , Técnico LSI-2001-01, 2001.

[38] Nuria Calvo Babío, "Análisis de la simulación dinámica como herramientade apoyo directivo a la gestión del talento en organizaciones de consultoría," *Revista Europea de Dirección y Economía de la Empresa*, vol. 18, no. 1, pp. 135-154, 2009.

[39] J. y Lardner, R. Arya, *Matemáticas aplicadas a la administración y a la economía*, Cuarta ed., Guillermo Trujano Mendoza, Ed. Mexico, Mexico: Pearson Education, 2002.

[40] Kniberg Henrik, "Scrum y XP Desde Las Trincheras," -, EEUU, Técnico 978-1-4303-2264-1, 2007.

[41] L., & Bartó, Godoy, "Validación y valoración de modelos en la Dinámica de Sistemas.," *Revista Argentina de Enseñanza de la Ingeniería*, 2002.

[42] Héctor MudarraTeruel, "Automatización de sistemas de desarrollo ágil Scrum: Team & Role -," Portal de Recursos Educativos Abiertos (REA), Memoria delProyecto de Fin de Carrera de Ingeniería Informática Bellaterra, Junio de 2010 2010.

[43] Tiago Keller Ferreira, "Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital.," UFSM, Informática/UFSM - Biblioteca Digital de Trabalhos de Graduação. , Estudo De Caso Em Empresa 2008.

[44] MaríaLaura. Citón, "Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad," Universidad de Mendoza - Facultad de Ingeniería, Mendoza, Trabajo Final de Carrera Ingeniería en Informática 2006.

[45] Luis Felipe El Shahili Gonzalez and Sharon Kornhauser López, *Burnout en el colectivo docente*, Primera edición ed. León, Mexico: Universidad EPCA, 2010.

[46] V. A Mahnic, "A case study on agile estimating and planning using scrum," *Electronics & Electrical Engineering*, vol. 111, no. 5, 2011.

[47] L Madeyski, *Test-Driven Development - An Empirical Evaluation of Agile Practice*.: Springer, 2010.

[48] Ken y Sutherland, Jeff. Schwaber, *Agile Software Development with Scrum*, Primera ed.: Prentice Hall, 2001.

[49] Diego Alberto Godoy and Kasiak Tamara, "Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP," in *Actas XVIII Congreso Argentino de Ciencias de la Computación*, 2012, p. 10.

[50] Firas Glaiel, "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics," Massachusetts Institute of Technology , Tésis de Máster 2012.

[51] J Aracil, *Dinámica de Sistemas*. Madrid, España: Alianza Editorial, 1997.

[52] Torrealdea J, *Dinámica de Sistemas. Elementos y Estructuras de un Modelo.Construyendo modelos*.: Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco, 2003.

[53] Ventana System Inc. (2013) Vensim. [Online]. http://www.vensim.com

[54] MaríaLaura. Citón, "Método Ágil lScrum Aplicado Al Desarrollo De Un Software De Trazabilidad," Universidad de Mendoza - Facultad de Ingeniería, Mendoza, Trabajo Final de Carrera Ingeniería en Informática 2006.

[55] V. A Mahnic, "case study on agile estimating and planning using scrum," *Electronics & Electrical Engineering*, vol. 111, no. 5, 2011.

[56] Héctor MudarraTeruel, "Automatización de sistemas de desarrollo ágil Scrum: Team & Role -," sitio Web temoa : Portal de Recursos Educativos Abiertos (REA), Memoria delProyecto de Fin de Carrera de Ingeniería Informática Bellaterra, Junio de 2010 2010.

Anexo I – Tabla de Variables y Funciones

En el presente Anexo se enumeran las Principales Variables del modelo, junto a las Funciones que determinan su comportamiento, no se consideran las variables de Nivel ya que el comportamiento de estas viene dado por los Flujos, y este comportamiento se puede observar en el Capítulo 6.

A continuación se presenta el listado de variables, que se encuentran ordenadas alfabéticamente por Subsistema y por el nombre de las variables.

Tabla 15 – Anexo I Tabla de Variables y Funciones

Variable	Función	Unidad	Tipo
Subsistema Adquisición De Experiencia			
auxExperienciaJuniorsPromos	auxPonderacionPromocionales*auxNivelProductividadJuniorsPromos	experiencia	auxiliar
auxExperienciaJuniorsContratados	auxNivelProductividadJuniors*auxPonderacionJuniors	experiencia	auxiliar
auxExperienciaSeniors	auxNivelProductividadSeniors*auxPonderacionSeniors	experiencia	auxiliar
auxNivelProductividadJuniors	0,1	Tasa	auxiliar
auxNivelProductividadJuniorsPromos	0,4	Tasa	auxiliar
auxNivelProductividadSeniors	0,9	Tasa	auxiliar
auxPonderacionJuniors	IF THEN ELSE(RHJUNIORSCONTRATADOS>0, RHJUNIORSCONTRATADOS*(1-(2.712^(tasaAprendizajeJrsContratados *Time))),0)	Tasa	auxiliar
auxPonderacionPromocionales	IF THEN ELSE(RHJUNIORSPROMOCIONALES>0, RHJUNIORSPROMOCIONALES*(1-(2.712^(-tasaAprendizajePromocionales *Time))), 0)	Tasa	auxiliar
auxPonderacionSeniors	IF THEN ELSE(RHSENIORS>0,RHSENIORS*(1-(2.712^(-tasaAprendizajeSeniors*Time))),0)	Tasa	auxiliar
EXPERIENCIATEAM	FljAdquisicionExperiencia	experiencia	nivel
tasaAprendizajeJrsContratados	0,1	Tasa	auxiliar
tasaAprendizajePromocionales	0,4	tasa	auxiliar

tasaAprendizajeSeniors	0,9	tasa	auxiliar
Subsistema Búsqueda Y Reclutamiento			
auxNroCandidatos	Variable según el proyecto	personas	auxiliar
auxTotalIntegrantes	Variable según el proyecto	personas	auxiliar
diferenciaIntegrantesTeam	Variable según el proyecto	personas	auxiliar
nroIdealIntegrantes	Variable según el proyecto	personas	auxiliar
porcentajeSeniorsBuscados	Variable según el proyecto	personas	auxiliar
RHSELECCIONADOS	fljPreseleccion-fljCandidatosSeniors-fljSeleccionCandidatos	personas	nivel
RHCANDIDATOSJUNIORS	fljSeleccionCandidatos-fljContratacionJuniors	personas	nivel
RHCANDIDATOSSENIORS	fljSeleccionCandidatos-fljContratacionSeniors	personas	nivel
Cansancio			
CANSANCIO	fljCansancio	cansancio	nivel
auxTasaCansancio	LOOKUP FORWARD(lkpCoeficienteCansancio 0, Time)	tasa	auxiliar
auxTasaCansancioDiario	LOOKUP FORWARD(lkpCoeficienteCansancioDiario ,(auxHorasDiarias+TAREASEXTRASDESARROLLAR))	tasa	auxiliar
diferenciaCansancio	auxTasaCansancioDiario+auxTasaCansancio-CANSANCIO-auxTasaCansancioDiario	cansancio	auxiliar
lkpCoeficienteCansancio	(0,0),(40,0.1),(80,0.15),(120,0.2),(160,0.25),(200,0.3),(240,0.35),(280,0.4),(320,0.45),(360,0.5),(400,0.52),(440,0.55),(480,0.57),(520,0.62),(600,0.7)	tasa	look up
lkpCoeficienteCansancioDiario	(1,0),(2,0.01),(3,0.02),(4,0.03),(5,0.04),(6,0.05),(7,0.06),(8,0.07),(9,0.08),(10,0.09),(11,0.1)	tasa	look up
Subsistema Control De Errores De Desarrollo			
auxTotalPruebas	pruebasPorTarea*TAREAS CODIFICADAS	pruebas	auxiliar
auxTotalPruebasIntegracion	((PRUEBASSINERROR/pruebasPorTarea)*pruebasIntegracionPorTarea)	pruebas	auxiliar
pruebasIntegracionPorTarea	Variable según el proyecto	pruebas	auxiliar
pruebasPorTarea	Variable según el proyecto	pruebas	auxiliar
tasaErrPorCansancio	CANSANCIO	tasa	auxiliar
tasaErrPorPresionPlazo	IF THEN ELSE(EXPERIENCIATEAM>0,(PRESIONPLAZO/EXPERIENCIATEA	tasa	auxiliar

	M),0)		
PRUEBASADISENAR	fljPruebasDisenadas-fljPruebasARealizar	pruebas	nivel
PRUEBASAREALIZAR	fljPruebasARealizar-fljErroresPorCodificacion-fljPruebasSinErrorCodifica	pruebas	nivel
PRUEBASCONERRDETECTADOS	fljErroresPorCodificacion-fljTareaConErroresDetectados	pruebas	nivel
PRUEBASSINERROR	fljPruebasSinErrorCodifica-fljPruebasIntegracion	Pruebas	nivel
Subsistema Desarrollo De Tareas			
auxAusenciaProduccion	IF THEN ELSE (INASISTENCIASNOACORDADAS<>0, INASISTENCIASNOACORDADAS* (auxVelocidadIdeal-(auxVelocidadIdeal/auxTotalIntegrantes)),0)	Puntos	auxiliar
auxTareasARecodificar	IF THEN ELSE(TAREASCONERRORES<fljTareasConErrDeCodificacion, fljTareasConErrDeCodificacion, 0)	tareas	auxiliar
auxTotalTareasConErroresDesarrollo	INTEGER(PRUEBASCONERRDETECTADOS/pruebasPorTarea +PRUEBASCONERRINTEGRACION/pruebasIntegracionPorTarea)	tareas	auxiliar
tareasPorPuntos	Variable según el proyecto	tareas	auxiliar
tasaAjusteVelocidad	Variable según el proyecto	tasa	auxiliar
TAREASCONERRORES	FljTareasConErrDeCodificacion	tareas	Nivel
TAREASCODIFICADAS	fljTareasCodificadas-fljTareasAProbar	tareas	nivel
TAREAS PENDIENTESCODIF	INTEGER(fljPlanificacionACodificacion+fljTareasConErrARecod+fljTareasExtras-fljTareasCodificadas)	tareas	nivel
TAREASPLANIFICADAS	fljTareasPendientes-fljPlanificacionACodificacion	tareas	nivel
Desarrollo Tareas Extras			
auxPeriodoTareaExtraNoPlanificada	IF THEN ELSE(probabilidadTareaExtraNoPlanificada>0, (FINAL TIME/3)*(1-probabilidadTareaExtraNoPlanificada),0)	horas	auxiliar
auxTareaExtraNoPlanificada	IF THEN ELSE(auxPeriodoTareaExtraNoPlanificada>5 :AND: Time>5, PULSE TRAIN(1, 1,auxPeriodoTareaExtraNoPlanificada	tarea	auxiliar

	, Time+1),0)		
auxTareasExtrasPorHacer	IF THEN ELSE(auxIniSprints=1,LOOKUP FORWARD(lkpTareasExtrasSprint, SPRINTSPLANIFICADOS),0)	tarea	auxiliar
lkpTareasExtrasSprint	Variable según el proyecto	tarea	look up
probabilidadTareaExtraNoPlanificada	Variable según el proyecto	tasa	auxiliar
TAREASEXTRASDESARROLLAR	fljTareasExtrasNoProgramadas+fljTareasExtrasDesarrollar-fljTareasExtrasATerminar	tareas	nivel
TAREASEXTRASREALIZADAS	fljTareasExtrasATerminar	tareas	nivel
Subsistema Horas Trabajadas			
auxHorasNormalesTrabajadas	Variable según el proyecto	horas	auxiliar
auxHsExtrasTrabajadas	Variable según el proyecto	horas	auxiliar
HORASTOTALESTRABAJADAS	FljHorasTrabajadasTotales	horas	nivel
HORASTRABAJADAS	fljHorasTrabajadas-fljHorasTrabajadasTotales	horas	nivel
lkpSprintDuracionEnHoras	Variable según el proyecto	horas	look up
lkpSprintHorasExtras	Variable según el proyecto	horas	Look Up
Subsistema Inasistencias			
auxInasistenciasNoAcordadas	IF THEN ELSE(auxPeriodoAsistenciaNoAcordada>0 :AND: Time > 10, PULSE TRAIN(1, 1,auxPeriodoAsistenciaNoAcordada , Time+1)*auxIntegrantesAusentes,0)	inasistencia	auxiliar
auxIntegrantesAusentes	INTEGER(RANDOM UNIFORM(1, auxTotalIntegrantes , 555555))	personas	auxiliar
auxPeriodoAsistenciaNoAcordada	IF THEN ELSE(probabilidadInasistenciaNoAcordada>0, INTEGER(FINAL TIME/4)*abs((probabilidadInasistenciaNoAcordada-1)),0)	horas	auxiliar
INASISTENCIASNOACORDADAS	fljProbabilidadInasistenciaNoAcordada-fljInasistenciaNoAcordada	inasistencia	nivel
TOTALINASISTENCIASNOACORDADAS	fljInasistenciaNoAcordada	inasistencia	nivel

probabilidadInasistenciaNoAcordada	Variable según el proyecto	tasa	auxiliar
Subsistema Planificación			
auxIniSprints	IF THEN ELSE(LOOKUP BACKWARD(lkpSprintsInicio,SPRINTSPLANIFICADOS)+1=Time,1,0)	sprint	auxiliar
auxSprintsTerminados	LOOKUP BACKWARD(lkpSprintsDuracionAcumulada,Time)	sprint	Look up
auxPuntosPorSprint	Variable según el proyecto	puntos	auxiliar
auxVelocidadIdeal	LOOKUP FORWARD(lkpVelocidadPorSprint,SPRINTSPLANIFICADOS)	puntos/hora	auxiliar
lkpPuntosPorSprint	Variable según el proyecto	puntos	look up
lkpSprintsDuraciónAcumulada	Variable según el proyecto	horas	look up
lkpSprintsInicio	Variable según el proyecto	horas	look up
lkpVelocidadPorSprint	Variable según el proyecto	puntos	look up
SPRINTSPLANIFICADOS	fljSprinsIdealesAterminar	Sprint	nivel
Subsistema Presión En El Plazo			
auxCoeficientePresionPlazo	(auxFinSprintReal*auxCoeficientePresionPlazo-PRESIONPLAZO)	tasa	auxiliar
auxFinSprintReal	IF THEN ELSE(FINAL TIME>Time,1-((FINAL TIME-Time)/FINAL TIME),1)	hora	auxiliar
diferenciaEnLaPresionPlazo	LOOKUP FORWARD(lkpPresionPlazo, auxFinSprintReal)	tasa	auxiliar
lkpPresionPlazo	(0,0.1),(0.1,0.1),(0.2,0.1),(0.3,0.15),(0.4,0.2),(0.5,0.25),(0.6,0.3),(0.7,0.4),(0.8,0.5),(0.9,0.5),(1,0.5)	tasa	look up
PRESIONPLAZO	fljPresionPlazo	presión	nivel
Subsistema Producción			
auxCalculaPuntos	IF THEN ELSE(fljPuntosAProcesar=0 :OR:PUNTOSPORHACER=0, fljPuntosAProcesar,IF THEN ELSE(PUNTOSPORHACER=auxPuntosPorSprint ,auxVelocidadIdeal, IF THEN ELSE(PUNTOSPORHACER+auxVelocidadIdeal > fljPuntosAProcesar,PUNTOSPORHACER ,fljPuntosAProcesar)))	puntos	auxiliar
auxCalculaPuntosPorHacer	IF THEN ELSE(auxIniSprints=1 ,0, IF THEN ELSE(((BURDOWNCHARTS)-	puntos	auxiliar

	auxVelocidadIdeal) >=auxVelocidadIdeal, (BURDOWNCHARTS-auxVelocidadIdeal), IF THEN ELSE(BURDOWNCHARTS=0 :OR: BURDOWNCHARTS=auxVelocidadIdeal,0, 0)))		
auxPuntos	IF THEN ELSE(auxIniSprints=1,auxPuntosPorSprint,0)	puntos	auxiliar
BURDOWNCHART	fljPuntosPorSprint-fljPuntosAprocesar	puntos	nivel
PUNTOSPORHACER	fljPuntosAprocesar-fljPuntosPorHacer	puntos	nivel
PUNTOSPORSPRINT	fljPuntos- fljPuntosPorSprint	puntos	nivel
Subsistema Promociones			
cantidadPasan	IF THEN ELSE(tasaAJuniors>0, IF THEN ELSE((RHJUNIORSCONTRATADOS *tasaAJuniors)>0.9, INTEGER(RHJUNIORSCONTRATADOS*tasaAJuniors), 0),0)	personas	auxiliar
demoraPaseAJuniors	Variable según el proyecto	horas	auxiliar
fljAbandonoJuniors	DELAY FIXED(IF THEN ELSE(tasaAbandonoJuniors>0, IF THEN ELSE(tiempoSalidaJuniors=Time,INTEGER(RHJUNIORSCONTRATADOS*tasaAbandonoJuniors),0),0), Time , 0)	personas	flujo
fljAbandonoJuniorsPromocionales	DELAY FIXED(IF THEN ELSE(tasaAbandonoPromocionables>0, IF THEN ELSE(tiempoSalidaPromocionales=Time, INTEGER(RHJUNIORSPROMOCIONALES*tasaAbandonoPromocionables), 0),0), Time , 0)	personas	Flujo
fljAbandonoSeniors	DELAY FIXED(IF THEN ELSE(tasaAbandonoSeniors>0 , IF THEN ELSE(tiempoSalidaSeniors=Time,INTEGER(RHSENIORS*tasaAbandonoSeniors),0),0), Time , 0)	personas	Flujo
fljJuniorsASeniors	INTEGER(RHJUNIORSPROMOCIONALES*tasaAvanceASenior)	personas	Flujo
fljPromocionDeJuniors	IF THEN ELSE(esperaEnPromocion>0,a1,0)	personas	Flujo

RHJUNIORSCONTRATADOS	fljContratacionJuniorsAux-fljAbandonoJuniors-fljPromocionales	personas	nivel
RHJUNIORSPROMOCIONALES	fljPromocionales-fljAbandonoJuniorsPromocionales-fljJuniorsASeniors	personas	nivel
RHSENIORS	fljJuniorsASeniors+fljContratacionSeniorsAux-fljAbandonoSeniors	personas	nivel
tasaAbandonoJuniors	Variable según el proyecto	tasa	auxiliar
tasaAbandonoPromocionables	Variable según el proyecto	tasa	auxiliar
tasaAvanceASenior	Variable según el proyecto	tasa	auxiliar
tiempoSalidaJuniors	Variable según el proyecto	horas	auxiliar
tiempoSalidaPromocionales	Variable según el proyecto	horas	auxiliar
tiempoSalidaSeniors	Variable según el proyecto	horas	auxiliar
Subsistema Pruebas De Integración			
tasaPruebasErrIntegracion	Variable según el proyecto	tasa	auxiliar
PRUEBASINTEGRACIONADISENAR	fljDisenoPruebasIntegracion - fljPruebasIntegracionARealizar	prueba	nivel
PRUEBASCONERRINTEGRACION	fljPruebasIntegracionARealizar-fljPruebasAIntegrar-fljPruebasConErrIntegracion	prueba	nivel
PRUEBASINTEGRACIONREALIZADAS	fljPruebasConErrIntegracion-fljPruebasErrIntegracionDetect	prueba	nivel
PRUEBASINTEGRADAS	fljPruebasAintegrar-fljPruebasIntegradas	prueba	nivel

Anexo II – Publicaciones en el marco de la Tesis

Simulando Proyectos de Desarrollo de Software Administrados con Scrum

Diego Alberto Godoy[a], Edgardo A. Belloni[b], Henry Kotynski[c], Héctor Dos Santos[d]
Eduardo O.Sosa[e]
Centro de Investigación en Tecnologías de la Información y Comunicaciones (C.I.T.I.C.)
Departamento de Ingeniería y Ciencias de la Producción-Universidad Gastón Dachary
Av. López y Planes 6519- Posadas, Misiones, Argentina. Teléfono: +54-376-4438677

[a] diegodoy@citic.ugd.edu.ar, [b] ebelloni@ugd.edu.ar, [c]henrykotynski@citic.ugd.edu.ar,
[d]hector2s@citic.ugd.edu.ar, [e]eduardo.sosa@citic.edu.ar

Resumen

El presente trabajo presenta una línea de investigación que tiene por objetivo general la elaboración e implementación de un modelo que permita simular la metodología de gestión de proyectos *Scrum*. Concretamente, se presenta aquí un modelo para simular uno de los componentes esenciales de la gestión de un proyecto desarrollado con Scrum: el Subsistema de Control de Errores de Tareas correspondiente a la fase denominada *Development Game*, propuesta por la metodología referida. El modelo presentado se ha validado con datos de tres proyectos reales. Adicionalmente, se han diseñado y ejecutado una serie de experimentos sobre el modelo propuesto y se ha realizado un Análisis de Sensibilidad de sus variables más importantes. El modelo construido sirve como ayuda a administradores de proyectos considerados novatos en la aplicación de la metodología referida, permitiéndoles conocer de antemano las consecuencias de sus decisiones.

Palabras claves: Administración de Proyectos de Desarrollo de Software; Scrum; Dinámica de Sistemas.

Contexto

El trabajo presentado en este artículo tiene como contexto marco el proyecto de investigación denominado "Simulación como Herramienta para la Mejora de los Procesos de Software", registrado actualmente en la Secretaría de Investigación y Desarrollo de la Universidad Gastón Dachary (UGD) [†] y radicado en el Centro de Investigación en Tecnologías de la Información y Comunicaciones de dicha universidad.

Entre las líneas con mayores resultados dentro del proyecto referido, se encuentran las de: (i) "Modelo de Simulación Dinámico de Gestión de Proyectos de Desarrollo de Software que utilizan Programación Extrema", y (ii) "Modelo de Simulación Dinámico de Gestión de Proyectos de Desarrollo de Software Bajo Scrum". Este artículo se enfoca en la presentación de la última línea aludida.

Introducción

La complejidad de los sistemas de actuales, los cambios repentinos en el contexto de estos sistemas y las modificaciones en los requerimientos del cliente una vez que se ha comenzado el desarrollo de un proyecto de software, generan un ambiente donde la planificación, el desarrollo, la administración y el control del mismo resultan difíciles de estimar o evaluar.

Este tipo de escenarios requiere de metodologías de desarrollo de software que permitan generar resultados rápidamente. Entre las metodologías con este tipo de características se encuentra *Scrum*, la cual fue aplicada por primera vez por Ken Schwaber y Jeff Sutherland, quienes la documentaron en detalle en suobra "Agile Software Development with Scrum" [1]. Esta metodología centra su atención en las actividades de gerencia basándose principalmente en una

[†] Mediante Res.Rectoral UGD Nº 04/I/12.-

planificación adaptativa y en el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo.

Frente a escenarios y requerimientos cambiantes, contar con una herramienta que permita simular la gestión de proyectos de desarrollo de software con *Scrum*, representa una alternativa interesante para que los administradores puedan evaluar el impacto de sus decisiones sobre la gestión en el desarrollo del proyecto, sin influir o poner en riesgo el proyecto real y sus recursos.

Actualmente existen diferentes trabajos y herramientas que permiten simular la administración de proyectos de software, entre ellos se puede citar a los siguientes: "Dynamics of Agile Software Development"[2], "Modeling Agile Software Maintenance Process Using Analytica lTheory of Project Investment"[3], "Modelo Dinámico de Simulación de Proyectos de Software con XP" [4] y [5], "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics" [6].

Estas herramientas presentan modelos basados en diferentes metodologías ágiles, en los cuales son propuestos y evaluados diferentes escenarios, algunos de ellos genéricos. No obstante, en ninguno de tales trabajos se presenta un estudio que contemple las fases y variables particulares de la metodología *Scrum*.

Objetivos Propuestos

Teniendo en cuenta lo planteado en la introducción en este trabajo se presentan los primeros avances en el diseño y la construcción de un Modelo de Simulación Dinámico basado el modelado sistémico que de soporte a la gestión de procesos de desarrollo de software que se basan en el uso de la metodología *Scrum*.

El modelo permite, que los administradores de proyectos, puedan evaluar qué impacto tendrán sus decisiones de gestión sobre el mismo, a lo largo del desarrollo. De esta manera el modelo refleja el efecto del uso de prácticas de Scrum en proyectos de desarrollo de software y para ello, su estructura permite simular el desarrollo de un Scrum completo a la vez. Es decir, dado que Scrum presenta un estilo de desarrollo iterativo e incremental, dentro de un proyecto se negocian con el cliente varias entregas o versiones, por lo que el modelo, está destinado a simular un Scrum por vez.

Construcción del Modelo

En el desarrollo y construcción del modelo se utilizó la herramienta CASE VenSim PLE 5.4c (Versión Académica) [7] seguido las etapas de la Metodología de Dinámica de Sistemas [8], de esta forma, en la Fase de Conceptualización se ha construido el Diagrama Causal, incluyendo las variables que representan un proyecto

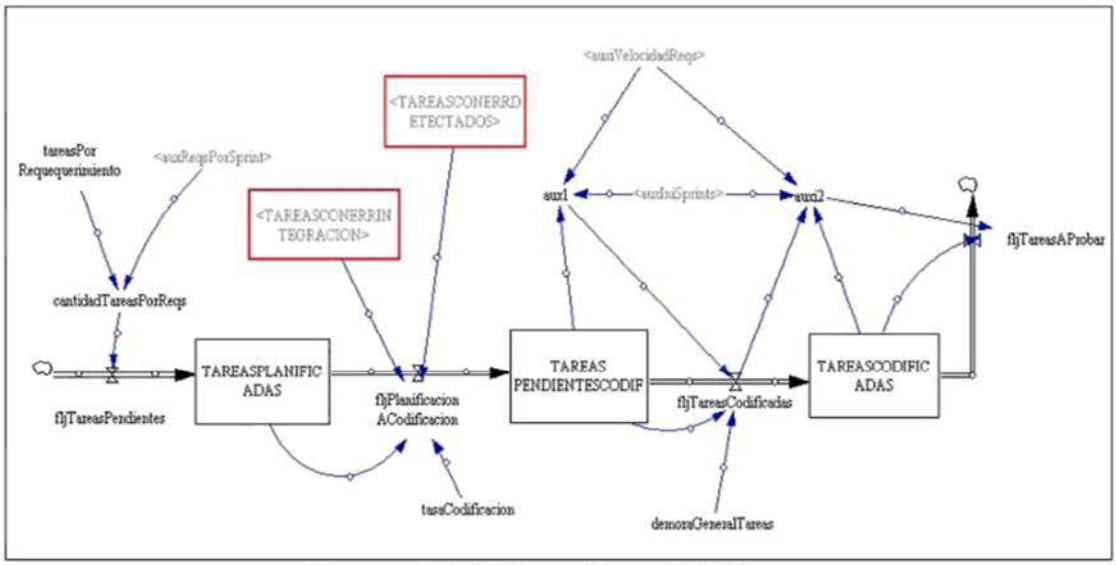


Diagrama1 - Sub-Sistema Desarrollo de Tareas

de software Scrum y sus interrelaciones. Luego, en la Fase de Formulación se ha traducido el Diagrama referido a un Diagrama de Forrester, el cual ha sido dividido en Subsistemas Conservativos de acuerdo con [9]. Finalmente, en la Fase de Evaluación se han realizado las corridas de validación y experimentales del modelo.

En el Diagrama 1, se presenta uno de los subsistemas más importantes del modelo: El Subsistema de Desarrollo de Tareas. En este Subsistema se muestra el proceso de planificación de las distintas tareas a codificar para satisfacer los distintos requerimientos del usuario seleccionados para cada *Sprint*. El subsistema se inicia a partir del número de requerimientos del usuario y una cantidad promedio de tareas necesarias para poder satisfacer a cada uno de estos. En la siguiente etapa, y en función del resultado de la primera etapa, a las tareas planificadas y pendientes de codificación se le suman aquellas tareas que tengan errores de Codificación y de Integración. En la última etapa se codifican o re-codifican las tareas provenientes de la etapa anterior. Las variables auxiliares Tasa de Codificación y Demora General de Tareas tienen respectivamente como función, seleccionar que tasa o porcentaje de las tareas planificadas se codificarán, y de agregar una demora temporal para el desarrollo de cada una de las tareas seleccionadas que finalmente se codifican y que luego serán probadas, generando de esta manera, Tareas con Errores de Integración o Con Errores de Programación. La variable auxiliar Velocidad de Requerimientos facilita la selección de la velocidad con la que se irán codificando las diferentes tareas planificadas, permitiendo esto, calcular la velocidad ideal a la cual podrá trabajar el Team a lo largo del sprint.

Validación

En la validación se utilizaron datos reales de tres Proyectos realizados con *Scrum*. El primero "Automatización de sistemas de desarrollo ágil Scrum: Team& Role", cuyos detalles se pueden ver en [10]. El segundo "Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital – Estudo De Caso Em Empresa De Software" [11]. El tercero de los casos utilizados Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad" [12].

Sprint	Requerimientos por Sprint	Tareas Promedio por Requerimiento
Plannig Meeting	1	1
1	8	8
2	8	8
3	12	12
4	9	9
5	9	9
6	7	7
7	10	9

Tabla 1 – Requerimientos Por Sprint del proyecto presentado en [10]

En el Gráfico1 se muestra el número de requerimientos de cada *Sprint* de acuerdo a los datos exhibidos en la segunda columna de la Tabla 1. Los valores de este gráfico sirven de inicio al subsistema del Diagrama 1 y son el resultado del número medio de tareas multiplicado por el número de requerimientos del usuario.

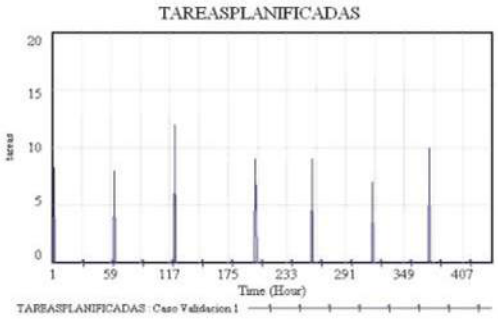


Gráfico1 - Tareas Planificadas por Sprint

En el Gráfico2 se puede observar el Burdown-Chart para todos los Sprints que componen el proyecto. El gráfico muestra como el número de tareas pendientes de codificación desciende desde los valores iniciales (Gráfico1), hasta llegar a cero, alcanzándose el nivel de Codificación planificado para cada Sprint (Gráfico2). Se aprecia también, como el nivel de codificación de Tareas se desarrolla de manera similar al presentado en el caso de validación [10].

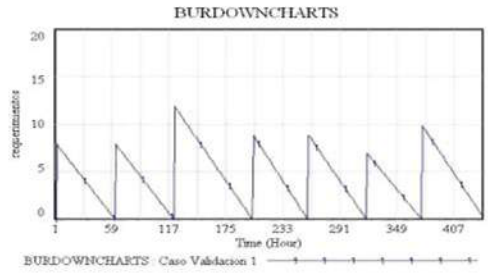


Gráfico2–Burdown-Chart Por Sprint

Experimentos realizados

En un experimento realizado basado en el primer caso de validación [10] se consideraron 63 requerimientos, distribuidos en 7 *Sprints*, resultando 9 requerimientos por *Sprint*. El número de integrantes no se modificó permaneciendo en 5 compuesto de la siguiente manera: 4 integrantes en el Team y 1 Stakeholder. La unidad de avance de tiempo elegida para la corrida fue de 1 hora. Considerando el número de requerimientos calculados para cada *Sprint* y asumiendo una tarea por cada requerimiento, se puede observar en el Gráfico4 los valores de tareas a planificar en cada *sprint*.

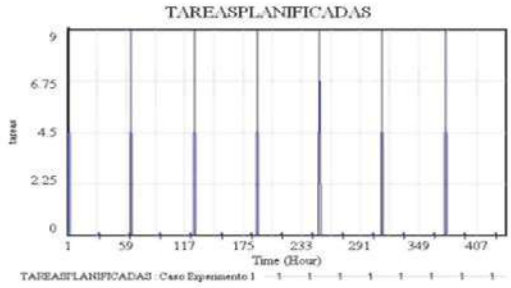


Gráfico3 - Tareas Planificadas Por Sprint

Por otra parte, en el Gráfico4 se puede observar el *Burdown-Chart* y como la cantidad de tareas pendientes descienden a medida que avanza el tiempo, y tomando el valor del número planificado (Gráfico3) de tareas para cada *Sprints* que componen el caso simulado en el experimento.

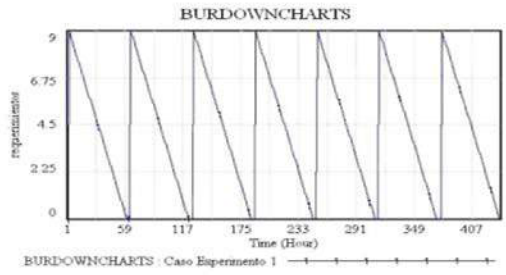


Gráfico4- Burdown-Chart Por Sprint

Resultados

En este trabajo se presentan avances en la construcción de un "Modelo de Simulación Dinámico de Gestión de Proyectos de Desarrollo de Software administrados con Scrum". Al enfocarse en las particularidades de *Scrum*, este trabajo se diferencia de otros en los que se modelan proyectos con Metodologías Tradicionales y en los que se modelan las generalidades de las Metodologías Agiles[2][3], o bien se modela la especificidad de una de éstas últimas como en[4]y [5].

El modelo presentado puede ser utilizado como herramienta para analizar el efecto que tiene el uso de prácticas de *Scrum* en proyectos de software. Para ofrecer flexibilidad, el modelo permite la alteración de valores durante su ejecución. Entre ellos los de: la cantidad de requerimientos del cliente; las fechas de comienzo y entrega de cada iteración; la cantidad de requerimientos a ser desarrollados en cada iteración; la cantidad de programadores; las horas extras agregadas por día, y el porcentaje de tiempo destinado a cada *Sprint*.

La validación realizada -con datos tomados de los proyectos [10], [11] y [12]como casos de "entrenamiento" se evalúa como positiva ya que el modelo construido se comportó de acuerdo con los datos de proyectos reales. Luego de validar el modelo se ha llegado a la conclusión de que el mismo cumple con sus objetivos y puede ser utilizado como herramienta para evaluar diferentes decisiones de gestión sobre proyectos de software desarrollados con *Scrum*.

Como trabajo futuro se espera avanzar, con la construcción de los demás subsistemas restantes, como por ejemplo: el de Desarrollo de Tareas; el de Recursos humanos y de Presión en el plazo, entre otros. Otra actividad prevista es la de compilar datos de proyectos de desarrollo de software en donde se utilizó *Scrum,* con el fin abarcar mayor cantidad proyectos. Adicionalmente, se prevé comparar la utilización de los simuladores de proyectos en *Scrum* y en XP[4] y [5], desarrollados anteriormente.

Formación de Recursos Humanos

El equipo de trabajo se encuentra formado por tres investigadores con distintos niveles de posgrado, un investigador con nivel de grado, y cuatro estudiantes en período de realización de trabajos finales de grado en el contexto de las carreras de Lic. en Sistemas de Información y de Ingeniería en Informática de la UGD. Actualmente, el número de tesinas de grado aprobadas en el contexto de este proyecto, es de uno, y otras dos en proceso de desarrollo.

Bibliografía

[1] Ken y Sutherland, Jeff. Schwaber, *Agile Software Development with Scrum*, Primera ed.: Prentice Hall, 2001.

[2] Kim E. Van, Kishore Sengupta, and Luk N. Van., "Dynamics of Agile Software Development," *International Conference of the System Dynamics Society*, 2009.

[3] Konga, Li, Liu y Chen, Jing Xiaoying, "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment," *International Conference on Advances in Engineering 2011*, 2011.

[4] Tamara Kasiak y Godoy Diego Alberto, "Simulación de Proyectos de Software desarrollados con XP," *XIV Workshop de Investigadores en Ciencias de la Computación.*, 2012.

[5] Godoy Diego Alberto y Kasiak Tamara., "Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP," in *Actas XVIII Congreso Argentino de Ciencias de la Computación*, 2012, p. 10.

[6] Firas Glaiel, "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics," Massachusetts Institute of Technology , Tésis de Máster 2012.

[7] Ventana System Inc. (2013) Vensim. [Online]. http://www.vensim.com

[8] J Aracil, *Dinámica de Sistemas*. Madrid, España: Alianza Editorial, 1997.

[9] Torrealdea J, *Dinámica de Sistemas. Elementos y Estructuras de un Modelo.Construyendo modelos.*: Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco, 2003.

[10] Héctor MudarraTeruel, "Automatización de sistemas de desarrollo ágil Scrum: Team & Role - ," sitio Web temoa : Portal de Recursos Educativos Abiertos (REA), Memoria delProyecto de Fin de Carrera de Ingeniería Informática Bellaterra, Junio de 2010 2010.

[11] Tiago Keller Ferreira, "Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital.," UFSM, Informática/UFSM - Biblioteca Digital de Trabalhos de Graduação. , Estudo De Caso Em Empresa 2008.

[12] MaríaLaura. Citón, "Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad," Universidad de Mendoza - Facultad de Ingeniería, Mendoza, Trabajo Final de Carrera Ingeniería en Informática 2006.

Evaluación de Alternativas de Gestión en Proyectos de Software Desarrollados con Scrum utilizando Dinámica de Sistemas

Diego A. Godoy, Edgardo Belloni, Eduardo O. Sosa, Henry Kotynski, Juan D. Benitez

Centro de Investigación en Tecnología de la Información y Comunicaciones (CITIC)
Departamento de Ingeniería y Ciencias de la Producción
Universidad Gastón Dachary
{diegodoy,ebelloni,eduardo.sosa,hkotynski,
juan.benitez}@citic.dachary.edu.ar

Resumen. En este trabajo se presenta un ejemplo de la utilización de un Modelo Dinámico de Simulación como herramienta de ayuda en la gestión de proyectos de desarrollo de software llevados a cabo con la metodología Scrum. Se describe la utilidad del simulador en la toma de decisiones en situaciones típicas que pueden ocurrir en este tipo de proyectos, como ser la incorporación de nuevas tareas durante el desarrollo. Con la ayuda de esta herramienta los Scrum Masters y miembros del Team podrán observar los resultados de cada decisión tomada y elegir la mejor alternativa para ser aplicada en un proyecto real.

Palabras Claves: Sistemas Dinámicos, Scrum, Proyectos de Software.

1 Introducción

Los cambios cada vez más rápidos en las tecnologías de soporte sistemas de software y los requerimientos del cliente una vez que se ha comenzado el desarrollo de un proyecto de software, generan un ambiente donde la planificación, el desarrollo, la administración y el seguimiento de los mismos resultan difíciles de estimar o evaluar.

Hoy en día este tipo de escenarios requiere de metodologías de desarrollo de software y gestión de proyectos que permitan generar resultados rápidamente y que los administradores de proyectos puedan usarlas de manera ágil. Entre las metodologías con este tipo de características se encuentra Scrum [1]. Esta metodología centra su atención en las actividades de gerenciamiento de proyectos basándose principalmente en una planificación adaptativa y en el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo.

Frente a escenarios y requerimientos cambiantes, contar con una herramienta que modele y permita simular la gestión de proyectos de desarrollo de software con Scrum, representa una alternativa interesante para que los administradores, en este caso los Scrum Master y el Team puedan evaluar el impacto de sus decisiones sobre

la gestión en el desarrollo del proyecto, sin influir o poner en riesgo el proyecto real y sus recursos. Asimismo, tal herramienta de simulación ayuda a construir consenso sobre las decisiones a tomar de manera objetiva.

Actualmente existen diferentes trabajos y herramientas que permiten simular la administración de proyectos de software, entre ellos se puede citar a los siguientes: "Dynamics of Agile Software Development" [2], "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment" [3], "ModeloDinámico de Simulación de Proyectos de Software con XP" [4] y[5], "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics" [6]. Estos trabajos presentan modelos basados en diferentes metodologías ágiles, en los cuales son propuestos y evaluados diferentes escenarios, algunos de ellos genéricos. No obstante, en ninguno de tales trabajos se presenta un estudio que contemple las fases, variables y roles, particulares de la metodología Scrum para el desarrollo de software.

En este contexto se hace necesario contar con una herramienta que asista a los Scrum Masters y los miembros del Team en la evaluación del impacto que tendría una o varias decisiones de gestión, compararlas entre sí y escoger la mejor de ellas para ser aplicada al proyecto real, sin comprometer el desarrollo del mismo. Cabe destacar que este trabajo se encuentra dentro de una línea de investigación presentada previamente en [7].

2 Construcción del Modelo

Para construir el modelo se han seguido las etapas de la Metodología de Dinámica de Sistemas [8], la cual consta de tres fases.

De esta forma, en la Fase de Conceptualización, se han identificado las variables intervinientes en el modelo que representa el sistema de gestión de un proyecto de software desarrollado con Scrum para luego construir el Diagrama de influencia que representa la estructura del sistema modelado. En la Figura 1 se puede ver un diagrama de influencias a nivel de Sub-sistemas y las relaciones causales que se manifiestan entre ellos.

Luego en la segunda fase, denominada Fase de Formulación se construyó el Diagrama de Forrester el cual constituye una traducción de las variables y relaciones del Diagrama de influencias a ecuaciones matemáticas, posibles de ser programadas y simuladas. Este Diagrama de Forrester, que se compone de más de 200 variables, fue además dividido en doce Subsistemas Conservativos de acuerdo con [9] y un subsistema de variables auxiliares y constantes, para facilitar su estudio y análisis. Los Sub-sistemas en que se ha dividido el modelo son:

- Planificación
- Producción
- Desarrollo de Tareas
- Pruebas de Desarrollo
- Pruebas de Integración
- Presión en el Plazo
- Promociones de R.H.
- Experiencia de R.H.
- Cansancio de R.H.
- Horas Trabajadas de R.H.
- Inasistencias de R.H.
- Variables auxiliares y Constantes

- Desarrollo de Tareas Extras

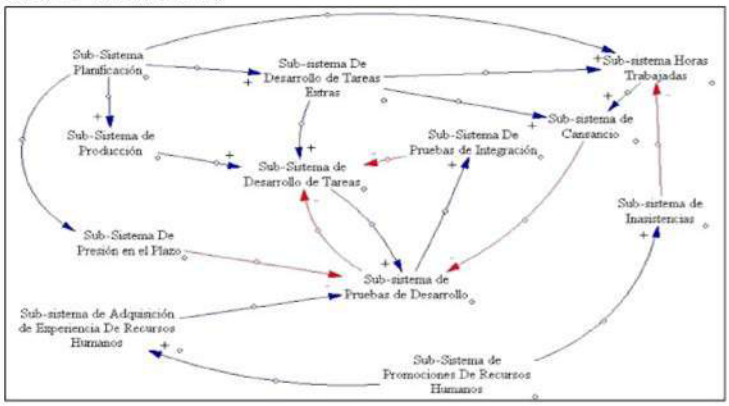


Figura1-Diagrama de relaciones causales entre Sub-sistemas.

A modo de ejemplo en este trabajo se presenta el diagrama de Forrester de Sub-sistema de Producción en la Figura 2 y se detallan las variables involucradas a continuación.

auxPuntos: Esta variable de tipo auxiliar permite determinar la cantidad de puntos que se desarrollarán en cada uno de los Sprints. Para poder determinar el contenido de esta variable la misma requiere de los valores de auxPuntosPorSprint y auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor establecido para dicho sprint.

auxCalculaPuntosPorHacer: Esta variable de tipo auxiliar permite determinar la velocidad ideal con la cual los puntos establecidos para cada Sprint deberán ir completándose. Para poder establecer el valor de esta variable la misma requiere de las variables: auxPuntosPorSprint y auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor de la velocidad ideal para dicho sprint.

auxCalculaPuntos: Esta variable de tipo auxiliar permite determinar la velocidad ideal con la cual los puntos establecidos para cada Sprint deberán ir completándose. Para poder establecer el valor de esta variable la misma requiere de las variables: auxPuntosPorSprint y auxIniSprint, así donde este determinado el inicio de un Sprint la variable tomará el valor de la velocidad ideal para dicho sprint.

PUNTOSPORSPRINT: Es la primera variable de nivel del sub-sistema, y en ella se almacenan los puntos que deberán ser desarrollados a lo largo de cada uno de los Sprints, y que provienen de la variable auxPuntos.

BURDOWNCHARTS : esta variable de nivel representa lo que en la realidad es el BurdownChart. La misma muestra cómo deberían desarrollarse los puntos previstos para cada Sprint, para esto el descuento de puntos se realiza en la medida que lo indique la variable auxCalculaPuntosPorHacer, y de esta manera el nivel presenta una forma de picos y caídas.

PUNTOSPORHACER: La tercera variable de nivel presente en el modelo representa la cantidad de puntos que deberían ir desarrollándose a lo largo del Sprint y en condiciones normales de trabajo, para esto el incremento de puntos se realiza en la

medida del valor que tome la variable auxCalculaPuntos. Al igual que el nivel anterior comportará en forma de picos.

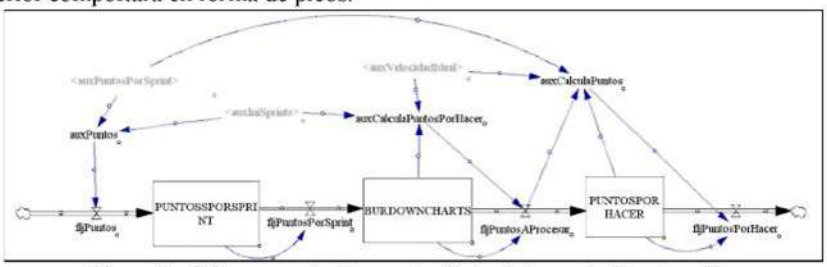


Figura2 - Diagrama de Forrester Sub-sistema de Producción

En última fase. la Fase de Evaluación. se han realizado corridas de validación y experimentales con el modelo. utilizando para ello casos reales y escenarios que representan situaciones frecuentes en proyectos gestionados con Scrum.

Se ha utilizado el Software VenSim PLE 5.4c (Versión Académica) [8] para la construcción de los modelos

A continuación en la sección 3 se presentará un caso de validación real del modelo y en la sección 4 experimentos realizados con el modelo.

3 Validación del Modelo

Para la validación del modelo se han utilizado datos de cuatro proyectos de software reales gestionados con Scrum siendo *Método Ágil lScrum Aplicado Al Desarrollo De Un Software De Trazabilidad*[11], *Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital*[12], *Case study on agile estimating and planning using scrum*[13] y *Automatización de sistemas de desarrollo ágil Scrum: Team & Role*[14].

En este trabajo se presentan los datos necesarios para validar si el modelo construido se comporta de acuerdo al proyecto descripto en [14]. Este proyecto se desarrolló en el marco de un convenio entre la Universidad Autónoma de Barcelona y la empresa UNIT4. La empresa especifica que el trabajo por parte de los programadores es a media jornada. una velocidad del 50% sería equiparable a una velocidad del 100% en un equipo a jornada completa. Para la validación se consideró que 1(uno) punto de historia es equivalente a 8(ocho) horas de trabajo. El detalle de las iteraciones se puede observar en la Tabla 1.

Tabla 1. Detalle de las iteraciones del proyecto presentado en[14]

Sprint	Requerimientos	Puntos Historias	Duración en Horas	Velocidad Ideal Estimada
1	9	18	80	0.2688
2	8	28	80	0.35
3	12	31	80	0.3875
4	9	21	80	0.2625
5	9	24	80	0.3
6	7	25	80	0.3125

7	10	25	80	0.3125
Total	64	172	560	

De acuerdo con los datos presentados en la Tabla 1, en la Figura 3 se puede observar la planificación puntos a desarrollar en cada uno de los Sprints.

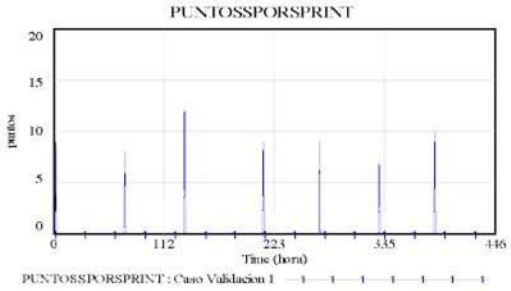


Figura 3 - Puntos Planificados Por Sprint

La velocidad de desarrollo de las tareas planificadas permitirá completar los puntos planificados para cada Sprint. En función de esta velocidad los Sprints se completaran en tiempo y forma, o bien habrá que quitar puntos de historia al mismo para poder cumplir con lo pactado con el Product Owner.

La cantidad de Sprints, los puntos a completar y la velocidad de trabajo de cada uno son valores que el Scrum Master junto al Team pueden estimar previamente al inicio de cada simulación de un proyecto.

En este caso de validación los Sprints son 7 (siete) y la cantidad de puntos en cada uno se puede observar en la Tabla 1. En la Figura 4 se puede observar el gráfico que representa el desarrollo ideal de puntos de historia planificados, presentados en la variable BURDOWNCHART, y de los puntos de historia pendientes a completar en la variable PUNTOSPORHACER.

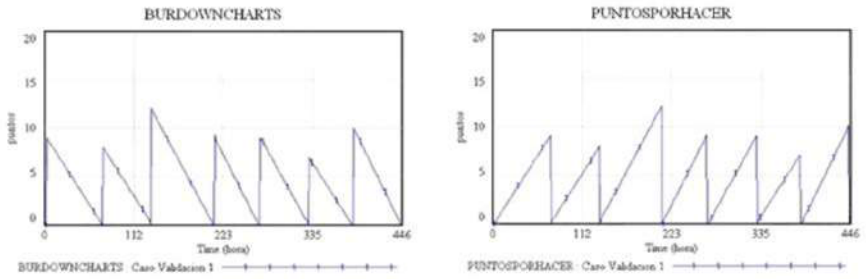


Figura 4 - Desarrollo ideal de los puntos y Puntos por hacer

De esta forma, y como puede verse en la Figura 4, la Cantidad de puntos por hacer desciende de manera ideal ya que este grafico se genera en la planificación previa al inicio de cada Sprint. Por otro lado, la cantidad de puntos por hacer asciende de cero al total planificado para dicho Sprint.

Esta situación se repite para todos los Sprints, ya que no se considera ningún tipo de tarea extra, inasistencia o abandono en el Team durante el desarrollo del proyecto.

En la medida que las tareas se planifican, y dadas las condiciones, dichas tareas se codifican sin inconvenientes y se completan en un 100% de acuerdo a lo planificado.

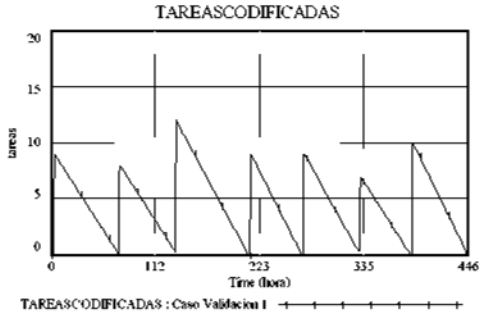


Figura 5 - Tareas Codificadas Por Sprint.

En la Figura 5 se puede ver también que tal como se ha planificado el comportamiento de la variable de nivel TAREASCODIFICADAS se comporta de manera similar al proyecto real validando él modelo.

4 Experimentos Realizados

El experimento presentado corresponde al desarrollo de un proyecto ficticio, pero que representa una situación real, gestionado con Scrum. El proyecto de prueba propuesto forma parte de una serie de 9 experimentos que fueron clasificados en tres niveles de dificultad (bajo, intermedio y avanzado), correspondiendo este experimento a un caso de nivel avanzado. En este caso la aparición tareas extras genera la necesidad de tomar decisiones ya con estas nuevas tareas no se puede cumplir el plazo del 248 hs del proyecto.

A continuación se detallan los principales valores de las variables utilizadas:

- Sprints: 6.
- Cantidad de Puntos de Historia por Sprint:
  - Sprint 1: 36
  - Sprint 2: 34
  - Sprint 3: 38
  - Sprint 4: 35
  - Sprint 5: 35
  - Sprint 6: 35
- Duración del proyecto: 248 horas.
- Cantidad de Programadores Novatos 4 y Expertos 2.
- Tareas Extras: Si.

Luego de ejecutar el modelo bajo las condiciones de inicio detalladas anteriormente, se observó que dada las condiciones de aparición de tareas extras la

velocidad ideal planificada no permitió cumplir con la codificación de las tareas planificadas, y por lo tanto tampoco con los puntos estimados al inicio de cada Sprint.

En la Figura 6 se presentan las Tareas planificadas inicialmente (TAREASPENDIENTES) y las Tareas extras por Sprint (TAREASEXTRASADESARROLLAR).

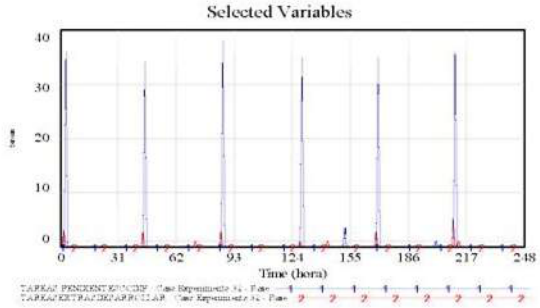


Figura 6 – 1) Tareas Extras Planificadas. 2) Tareas Extras No Planificadas

En la Figura 7 se observan superpuestas las corridas donde se consideró el proyecto sin tareas extras y con tareas extras.

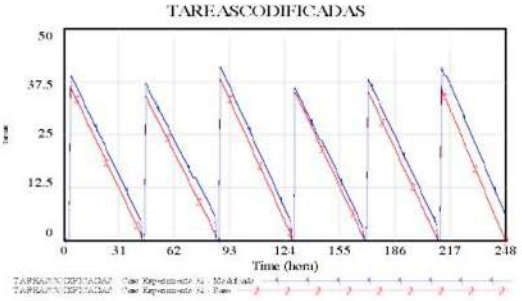


Figura 7 - 1) Proyecto sin tareas extras. 2) Proyecto con Tareas Extras

Se observa que en la corrida donde se consideró el proyecto con tareas extras, no termina en el tiempo planificado, sino que aún quedan pendientes tareas al momento de iniciarse el siguiente Sprint y en el último Sprint al finalizar el tiempo planificado del proyecto la situación es similar no completándose el proyecto en las horas planificadas.

Dada esta situación se presentan dos alternativas para solucionar el inconveniente de la aparición de tareas extras no planificadas, la primera de ellas sería incorporar horas extras a las horas de trabajo por día, hasta que el problema sea corregido, o bien la segunda alternativa es aumentar la velocidad de desarrollo de las tareas o considerar algún factor de ajuste a la velocidad ideal estimada. En la siguiente figura 8 se muestra el resultado de las tareas codificadas luego de realizar un ajuste uniforme de un 5% en la velocidad de desarrollo de cada uno de los Sprints. Logrando finalizar

la codificación de las tareas antes del tiempo previsto, pero quedando horas donde no se realizaron actividades.

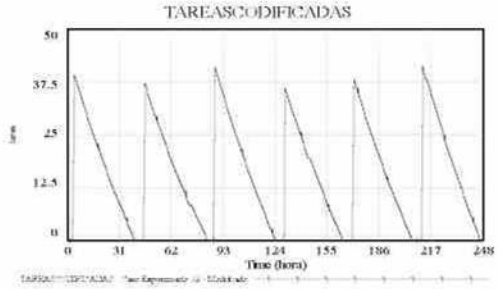


Figura 8– Sprints luego de ajustar la velocidad del desarrollo

En la Figura 9 se muestra una solución alternativa, resultando en un aumento de 26 horas en la duración total del proyecto, pero logrando alcanzar el final de cada sprint con el total de tareas codificadas.

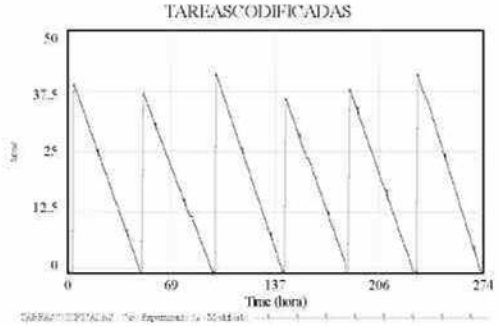


Figura 9–Sprints luego de aumentar en 10 horas la duración del proyecto.

En este caso el Scrum Master junto con el Team podrán optar por la primera alternativa, si deciden que pueden tener horas de programadores ociosos (o asignarlos a otros proyectos) o extender el tiempo del proyecto en el caso de la segunda alternativa.

5 Conclusiones

En el presente trabajo se ha presentado un caso de validación y un experimento realizado con un "Modelo Dinámico de Simulación para Proyectos de Software que utilizan Scrum". El modelo permite a los Scrum Masters y el Team analizar el efecto del uso conjunto de la metodología Scrum, Bloques de Tiempo, Artefactos y Reglas en la gestión de los mismos en diferentes escenarios. El modelo ofrece flexibilidad de modificar los valores de los parámetros tanto previamente al inicio de la simulación como al momento de la ejecución de la misma. Dentro de los parámetros que se

pueden establecer previo al inicio de cada simulación se encuentran: la duración y la velocidad de cada Sprint, la velocidad estimada de desarrollo de las tareas, Factores de Cansancio, de Presión en el plazo, Cantidad de integrantes del Team según su experiencia en la metodología y las tareas extras que se prevén puedan surgir. A través de la modificación de valores de los parámetros durante su ejecución el usuario puede establecer o modificar la cantidad de integrantes del Team que abandonan el proyecto, clasificar al Team mediante la asociación de estos a su experiencia en Scrum en Juniors o Expertos, cambiar la cantidad de horas estimadas de duración del proyecto, generar horas extras e inasistencia de los integrantes de manera determinística o pseudoaleatoria, entre otros.

De esta forma, luego de validar y experimentar con el modelo, se ha llegado a la conclusión de que el mismo puede ser utilizado como herramienta para evaluar el impacto de diferentes decisiones de gestión como la aparición tareas extras, presentado en la sección 5.

6 Trabajos Futuros

Como trabajos futuros se planea Adicionar otros sub-sistemas que permitan ampliar el ámbito del modelo, lo que permitiría una herramienta que facilite aún más las tareas previas al inicio del proyecto al Scrum Master y al Team. Dentro de estos sub-sistemas podrían nombrarse: cálculo de costos, comunicación en el Team, Intercambio de Roles, Adquisición de Experiencia, por nombrar solamente algunos. Así también, se pretende avanzar con la construcción de modelos similares para otras metodologías consideradas ágiles como Cristal Clear y Crystal Orange [15] [16] o incluir prácticas como por ejemplo Test Driven Development [17].

Por otra parte se planea utilizar este simulador en cátedras relacionadas con la Ingeniería de Software, con el fin de entrenar a futuros Scrum Masters y Teams.

Finalmente, es preciso construir bases de datos de Proyectos Scrum que contengan datos de proyectos de software reales llevados a cabo, ya que actualmente resulta difícil contar con datos post mortem de proyectos gestionados con métodos ágiles.

7 Bibliografía

[1] Ken y Sutherland, Jeff. Schwaber, *Agile Software Development with Scrum*, Primera ed.: Prentice Hall, 2001.

[2] Kim E. Van, Kishore Sengupta, and Luk N. Van., "Dynamics of Agile Software Development," 2009.

[3] Xiaoying Konga, Li Liu, and Chen Jing, "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment," *International Conference on Advances in Engineering 2011*, 2011.

[4] Tamara Kasiak and Diego Alberto Godoy, "Simulación de Proyectos de Software desarrollados con XP," *XIV Workshop de Investigadores en Ciencias de la Computación.*, 2012.

[5] Diego Alberto Godoy and Kasiak Tamara, "Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP," in *Actas XVIII Congreso*

Argentino de Ciencias de la Computación, 2012, p. 10.

[6] Firas Glaiel, "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics," Massachusetts Institute of Technology , Tésis de Máster 2012.

[7] Diego Alberto Godoy, Edgardo A. Belloni, Henry Kotynski, Hector H Dos Santos, and Eduardo Omar Sosa, "Simulando Proyectos de Desarrollo deSoftware Administrados con Scrum," in *XVI Workshop de Investigadores en Ciencias de la Computación RedUNCI*, Ushuaia, 2014.

[8] J Aracil, *Dinámica de Sistemas*. Madrid, España: Alianza Editorial, 1997.

[9] Torrealdea J, *Dinámica de Sistemas. Elementos y Estructuras de un Modelo.Construyendo modelos.*: Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco, 2003.

[10] Ventana System Inc. (2013) Vensim. [Online]. http://www.vensim.com

[11] MaríaLaura. Citón, "Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad," Universidad de Mendoza - Facultad de Ingeniería, Mendoza, Trabajo Final de Carrera Ingeniería en Informática 2006.

[12] Tiago Keller Ferreira, "Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital.," UFSM, Informática/UFSM - Biblioteca Digital de Trabalhos de Graduação. , Estudo De Caso Em Empresa 2008.

[13] V. A Mahnic, "case study on agile estimating and planning using scrum," *Electronics & Electrical Engineering*, vol. 111, no. 5, 2011.

[14] Héctor MudarraTeruel, "Automatización de sistemas de desarrollo ágil Scrum: Team & Role -," sitio Web temoa : Portal de Recursos Educativos Abiertos (REA), Memoria delProyecto de Fin de Carrera de Ingeniería Informática Bellaterra, Junio de 2010 2010.

[15] Alistair Cockburn, *Crystal Clear, A Human-Powered Methodology for Small Teams.*: Addison-Wesley Professional, 2004.

[16] Cockburn, Alistair, *Agile Software Development: The Cooperative Game.*: Addison-Wesley Professional, 2006.

[17] L Madeyski, *Test-Driven Development - An Empirical Evaluation of Agile Practice.*: Springer, 2010.

Simulación Dinámica de Gestión de Tareas en Proyectos Desarrollados Con Scrum

Diego Alberto Godoy, Henry Kotynski, Edgardo Belloni, Eduardo O. Sosa,
Centro de Investigación en Tecnología de la Información y Comunicaciones
Departamento de Ingeniería y Ciencias de la Producción
Universidad Gastón Dachary
Av. López y Planes 6519- Posadas, Misiones, Argentina. Teléfono: +54-376-4438677
{diegodoy hkotynski,ebelloni,eduardo.sosa}@citic.dachary.edu.ar

Resumen

En este trabajo se presenta el subsistema de gestión de tareas como parte de un "Modelo de Simulación Dinámico de Gestión de Proyectos de Desarrollo de Software Bajo Scrum". Se presenta un caso de validación y un caso de experimentación de situaciones reales en el cual se pueden ver situaciones de utilidad del modelo. Con la ayuda de esta herramienta los Scrum Masters y miembros del Team podrán observar los resultados de cada decisión tomada y elegir la mejor alternativa para ser aplicada en un proyecto real.

1. Introducción

La complejidad actual de las organizaciones y sus sistemas de software están sujetos a constantes cambios tanto tecnológicos como de requisitos. Esto genera un entorno donde la planificación, el desarrollo, la administración y el seguimiento del software resultan difíciles de estimar o evaluar.

Es por ello que este tipo de escenarios requiere de metodologías de desarrollo de software y gestión de proyectos que permitan generar resultados rápidamente y que los encargados de gestionar proyectos de software puedan utilizarlas de la mejor manera para mantener bajo control el proyecto. Una de las metodologías con este tipo de características es Scrum [1], la cual hace énfasis en las tareas de gestión de proyectos utilizando la planificación adaptativa y el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo.

En escenarios y requerimientos con constantes cambios, sería interesante contar con una herramienta que permita simular proyectos de desarrollo de software gestionados con Scrum, que permita a los Scrum Master y el Team evaluar el impacto de sus decisiones sobre la gestión en el desarrollo de proyectos, sin influir o poner en riesgo el proyecto real y sus recursos. De la misma manera, tal herramienta de simulación ayuda a construir consenso sobre las decisiones a tomar de manera objetiva y no destructiva.

Si bien actualmente existen diferentes trabajos y herramientas que permiten simular la administración de proyectos de software, entre los cuales se pueden citar a los siguientes: "Dynamics of Agile Software Development" [2], "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment" [3], "Modelo Dinámico de Simulación de Proyectos de Software con XP" [4] y[5], "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics" [6],los mismos presentan modelos basados en diferentes metodologías ágiles, en los cuales son propuestos y evaluados diferentes escenarios, algunos de ellos genéricos. No obstante, en ninguno de tales trabajos se presenta un estudio que contemple las fases, variables y roles, particulares de la metodología Scrum para el desarrollo de software. En este caso se presentará el subsistema de gestión de gestión de tareas como parte de un "Modelo de Simulación Dinámico de Gestión de Proyectos de Desarrollo de Software Bajo Scrum". Cabe destacar que este trabajo se encuentra dentro de una línea de investigación presentada previamente en [7].

El presente trabajo se estructura de la siguiente manera: en la sección 2 se explicará brevemente la metodología de dinámica de sistemas. En la sección 3 se dará una descripción de la construcción del modelo. En la sección 4 se presentará un proyecto real utilizado como caso de validación del modelo y en la sección 5, se realizaran pruebas de experimentación con el modelo de situaciones problemáticas en los proyectos de software desarrollados siguiendo Scrum y se evaluaran dos

posibles alternativas para atacarlos. Por último se presentaran las conclusiones y trabajos futuros.

2. Metodología de Dinámica de Sistemas.

El modelo se construyó basándose en la Metodología de Dinámica de Sistemas [8], la cual consta de tres fases.

1. Fase de Conceptualización, se han identificado las variables intervinientes en el modelo que representa el sistema de gestión de un proyecto de software desarrollado con Scrum, para luego construir el Diagrama de influencia que representa la estructura del sistema modelado.
2. Luego, en la segunda fase, denominada Fase de Formulación se construyó el Diagrama de Forrester el cual constituye una traducción de las variables y relaciones del Diagrama de influencias a ecuaciones matemáticas, posibles de ser programadas y simuladas. Este Diagrama de Forrester, que se compone de más de 200 variables, fue además dividido en doce Subsistemas Conservativos de acuerdo con [9] y un subsistema de variables auxiliares y constantes, para facilitar su estudio y análisis.
3. En última fase, la Fase de Evaluación, se han realizado corridas de validación y experimentales con el modelo, utilizando para ello casos reales y escenarios que representan situaciones frecuentes en proyectos gestionados con Scrum.

Para la construcción del modelo se ha utilizado el Software VenSim PLE 5.4c (Versión Académica) [10].

3. Construcción de Modelo.

Para la construcción del modelo se utilizado la metodología presentada en la sección 2.

Los Sub-sistemas en que se ha dividido el modelo son:

- Planificación
- Producción
- Desarrollo de Tareas
- Pruebas de Desarrollo
- Pruebas de Integración
- Presión en el Plazo
- Desarrollo de Tareas Extras
- Promociones de R.H.
- Experiencia de R.H.
- Cansancio de R.H.
- Horas Trabajadas de R.H.
- Inasistencias de R.H.
- Variables auxiliares y Constantes

A modo de ejemplo en este trabajo se presenta el diagrama de Forrester de Sub-sistema de Gestión de Tareas en la Figura 1 y se detallan las variables involucradas a continuación.

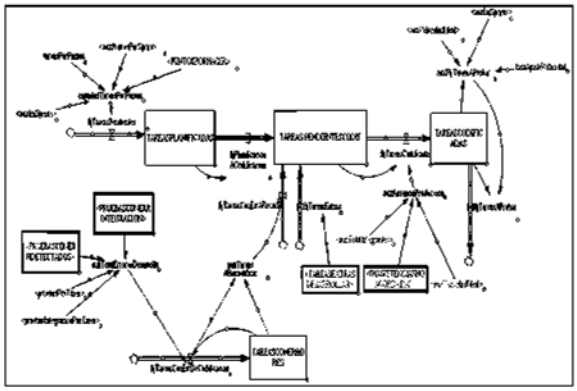


Figura 1 –Sub-sistema de Desarrollo de Tareas

3.1. Variables del Subsistema de Desarrollo de tareas

tareasPorPuntos: esta variable permite establecer la cantidad de tareas por puntos de historia que el Scrum Master estime o considere oportunas dada la complejidad de las tareas a desarrollar. Unidad: tareas

cantidadTareasPorPuntos: representa la cantidad total de tareas que se deben realizar para poder completar cada uno de los puntos de historia establecidos por Sprint. El valor de esta variable surge del producto entre el valor asignado a tareasPorPuntos y auxPuntosPorSprint. Dicho producto se efectuará solamente si PUNTOSPORHACER es mayor a cero. Unidad: tareas.

TAREASPLANIFICADAS: variable de nivel que representa la cantidad total de tareas a desarrollar en los diferentes Sprints. El valor de esta lo determina cantidadTareasPorPuntos. Unidad: tareas.

TAREASPENDIENTESCODIF: este nivel representa la cantidad de tareas pendientes de codificación en cada uno de los Sprints. El valor inicial está determinado por las tareas originales que se planificaron, y se ve modificado de acuerdo a la cantidad de Tareas con Errores que pudieran surgir y de las Tareas Extras. Unidad: tareas,

TAREASCODIFICADAS: almacena la cantidad de tareas que se fueron codificando a lo largo del desarrollo del Sprint. El desarrollo presenta la forma general de un diente de sierra, con puntos altos y descensos lineales, que se ven alterados por la ausencia no planificada de los integrantes del Team o por la aparición de Tareas no planificadas. Unidad: tareas.

auxAusenciaProduccion: esta variable auxiliar permite establecer en cuanto disminuirá la velocidad normal de producción diaria en función de la cantidad de ausencias no acordadas de los integrantes, de cuantos integrantes no asistieron al Sprint diario y de la velocidad de trabajo estimada inicialmente. Unidad: puntos.

tasaAjusteVelocidad: variable que permite ajustar la velocidad ideal establecida en el sub-sistema de Planificación. Ya que el número de tareas en relación a los puntos de historia no siempre es una tarea a un punto, se hace necesario modificar la velocidad de codificación para que los tiempos establecidos sean respetados. Unidad: tasa.

TAREASCONERRORES: variable de Nivel que acumula la totalidad de las tareas con errores sean de Integración o de Codificación. Su valor surge de la suma de la cantidad de tareas con errores de codificación dividido la cantidad de pruebas, más la cantidad de tareas con errores de integración divido la cantidad de pruebas de integración. Unidad: tareas.

auxTotalErroresDesarrollo: variable auxiliar donde se calcula el número de tareas con errores. Su valor se establece a partir de la suma de dos valores, ambos surgidos de dividir, el total de pruebas con errores de integración sobre el número de pruebas de integración, y del total de errores por codificación detectados sobre el número de pruebas realizadas. Unidad: errores

auxTareasARecodificar: variable auxiliar que permite determinar el número total de tareas con error existentes, generando picos que representan el número de tareas que surgen a lo largo del proyecto. Unidad: tareas.

4. Validación del Modelo

Para la validación se han utilizado datos de los siguientes proyectos de software reales gestionados con Scrum. Siendo estos los siguientes: Método ÁgilScrum Aplicado Al Desarrollo De Un Software De Trazabilidad[11], Aplicação Do Processo Ágil De GerenciamentoScrum No Desenvolvimento De UmJogo Digital[12], Case studyon agile estimating and planning using scrum [13] y Automatización de sistemas de desarrollo ágil Scrum: Team & Role[14].

Como ejemplo de validación en[14] se desarrolló en el marco de un convenio entre la Universidad Autónoma de Barcelona y la empresa UNIT4. El trabajo por parte de los programadores es a media jornada, una velocidad del 50% sería equiparable a una velocidad del 100% en un equipo a jornada completa. Para la validación se consideró que 1(uno) punto de historia es equivalente a 8(ocho) horas de trabajo. El detalle de las iteraciones se puede observar en la Tabla 1.

Tabla 1.Iteraciones del proyecto presentado en[14]

Sprint	Requerimientos	Puntos Historias	Duración en Horas	Velocidad Ideal Estimada
1	9	18	80	0.2688
2	8	28	80	0.35
3	12	31	80	0.3875
4	9	21	80	0.2625
5	9	24	80	0.3
6	7	25	80	0.3125
7	10	25	80	0.3125
Total	64	172	560	

De acuerdo con los datos presentados en la Tabla 1, en la Figura 2 se puede observar la planificación puntos a desarrollar en cada uno de los Sprints.

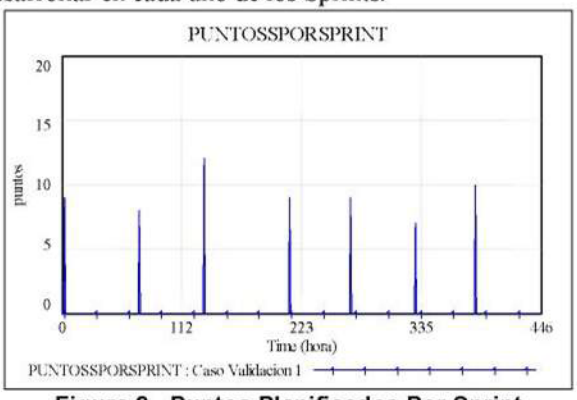


Figura 2 - Puntos Planificados Por Sprint

La velocidad de desarrollo de las tareas planificadas permitirá completar los puntos planificados para cada Sprint. En función de esta velocidad, los Sprints se completaran en tiempo y forma, o bien habrá que quitar puntos de historia para cumplir con el ProductOwner.

La cantidad de Sprints, los puntos a completar y la velocidad de trabajo de cada uno, son valores que el Scrum Master junto al Team pueden estimar previamente al inicio de cada simulación de un proyecto. En este caso de validación los Sprints son siete y la cantidad de puntos en cada uno se puede observar en la Tabla 1. En la Figura 3 se puede observar el gráfico que representa el desarrollo ideal de puntos de historia planificados, presentados en la variable BURDOWNCHART, y de los puntos de historia pendientes a completar en la variable PUNTOSPORHACER.

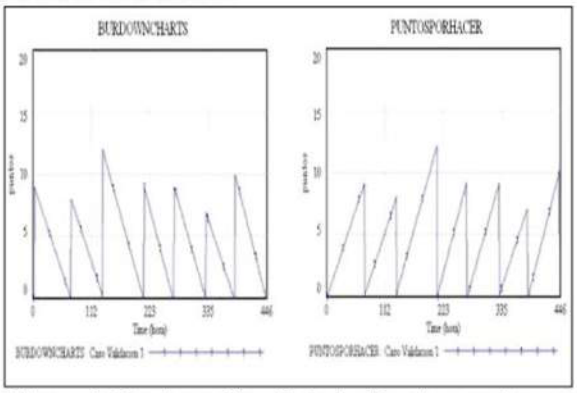


Figura 3–BurdownChart ideal y Puntos por hacer

De esta forma, y como puede verse en la Figura 3, la cantidad de puntos por hacer desciende de manera ideal ya que este grafico se genera en la planificación previa al inicio de cada Sprint. Por otro lado, la cantidad de puntos

por hacer asciende de cero al total planificado para dicho Sprint.

Esta situación se repite para todos los Sprints, ya que no se considera ningún tipo de tarea extra, inasistencia o abandono en el Team durante el desarrollo del proyecto.

En la medida que las tareas se planifican, y dadas las condiciones mencionadas, dichas tareas se codifican sin inconvenientes y se completan en un 100% de acuerdo a lo planificado.

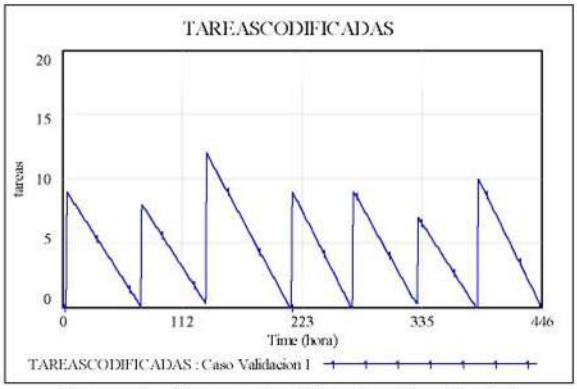


Figura 4 - Tareas Codificadas Por Sprint.

En la Figura 4 se puede ver también que tal como se ha planificado el comportamiento de la variable de nivel TAREASCODIFICADAS se comporta de manera similar al proyecto real validando él modelo.

5. Evaluación del Modelo

En este caso de evaluación del modelo se consideran situaciones donde ciertos parámetros considerados críticos toman valores diferentes a fin de probar el modelo ante situaciones extremas.

5.1. Parámetros Experimento

La Tabla 2 presenta los parámetros generales para el experimento.

Variable	Valor	Observación
Horas Diarias	7	
Días semana	5	
Hs. Extra por Semana	1	
HorasTotalesporSemana	35	
Horas Totales Normales	210	
Horas Totales Extras	6	
Horas Totales Proyecto	216	
Sprints	6	
Puntos de Historia	214	
PruebasPorTarea	2	
Velocidad Prueba	0	
Inasistencias Pactadas	3	
Inasistencias Imprevistas	5	15%
Factor Dedicación Sprint	70,0%	
Factor Dedicación Diario	80,0%	
Días Hombre Ideal	105	
Días Hombre Real	94,5	
Team	6	
Errores por Presión en el plazo	Si	
Errores de Integración	Si	15%
Promociones en el Team	No	
Abandonos en el Team	No	
Tareas Extras	Si	
Tareas Extras No planificadas	Si	20%

La Tabla 3 presenta los valores iniciales utilizados para la corrida.

Tabla3–Experimento - Parámetros Iniciales

Sprint	(a)	(b)	(c)	(d)	(e)
1	36	1,000	35	0	1
2	34	1,000	35	35	1
3	38	1,000	35	70	1
4	35	0,900	35	105	1
5	35	0,900	35	140	1
6	36	0,990	35	175	1
Total	**214**		**210**		**6**
(f)	**(g)**	**(h)**	**(i)**	**(j)**	**(k)**
36	2	38	1	1	0
36	3	37	0	0	36
36	1	39	0	1	72
36	2	37	0	0	108
36	3	38	1	0	144
36	2	38	0	1	180
216	**13**	**227**	**2**	**3**	

Tabla 2–Parámetros Generales del Experimento

El detalle de cada columna es el siguiente:

a) Puntos Historia: Determinado por el Team.
b) Duración planificada en horas: Surge de multiplicar las horas normales de trabajo por la cantidad de días semanales de trabajo.

c) Velocidad Estimada: Inicialmente surge de la división de (a) sobre (b). Luego se realizan ajustes para generar el BurdownChart
d) Inicio Planificado: Suma acumulada de los diferentes valores de (b).
e) horas extras: Determinadas según la necesidad del Team. Expresa las horas extras semanales.
f) Duración con Hs. Extras: Surge de la suma de (b) más (e).
g) Tareas Extras: Determinadas por el Scrum Master en función de la necesidad del Team.
h) Puntos con Tareas Extras: Surge de sumar (a) más (g).
i) Tareas Extras No Planificadas: Generado Aleatoriamente.
j) Ausencias No acordadas: Generado Aleatoriamente.
k) Inicio Recalculado: Suma acumulada de los diferentes valores de (f).

5.2. Observaciones

En este caso de experimentación y a fin de poder ejemplificar las situaciones modeladas, se presentan dos escenarios para las corridas, una donde se incluyen las tareas extras planificadas y las tareas a recodificar, y otro que presenta los datos base iniciales presentados en la Tabla 3.

Además se presentan situaciones donde como consecuencia de la inclusión de tareas se debió recalcular porcentualmente la velocidad de codificación estimada inicialmente.

5.3. Resultado del experimento

El inicio de los Sprints para este escenario se observan en la Figura 5.

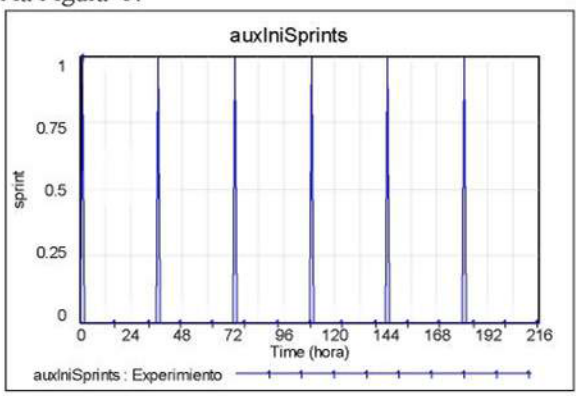


Figura 5 – Inicio de los Sprints

Otro de los gráficos relevantes para el modelo presentando es el de los puntos planificados por Sprint, para actual Experimento dichos puntos se presentan en la Tabla 3, y se muestran en la Figura 6.

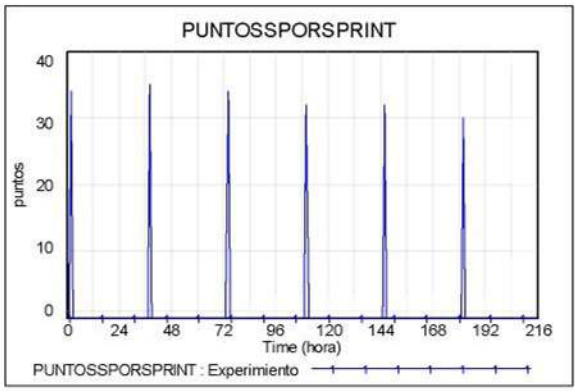


Figura 6– Puntos Planificados por Sprint

En función de los puntos planificados y la velocidad estimada de desarrollo se genera el BurdownChart, que muestra un avance ideal del proyecto y del desarrollo de los puntos planificados. En la Figura 7 se observa el comportamiento de dicha variable de nivel.

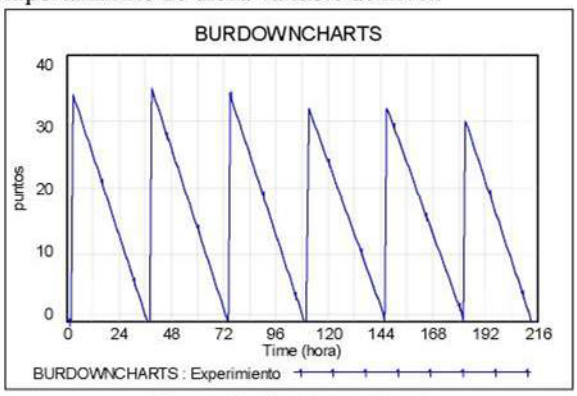


Figura 7– BurdownChart

Como se mencionó anteriormente la velocidad de avance permite completar las tareas planificadas y acordadas por el Team. En la Figura 8 se presentan las velocidades para cada uno de los Sprints del presente caso.

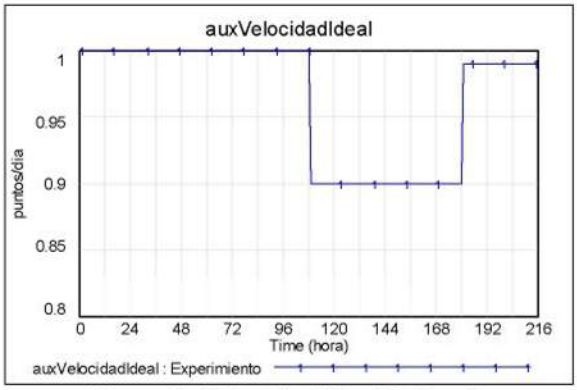


Figura 8– Velocidad Ideal Estimada

En función de la velocidad estimada y de los puntos de historia seleccionados para cada sprint, se genera un gráfico que muestra el avance ideal de cómo se deberían ir completando los puntos en cada uno de los Sprints. En la Figura 9 se observa este comportamiento descripto y representado por la variable de nivel PUNTOSPORHACER.

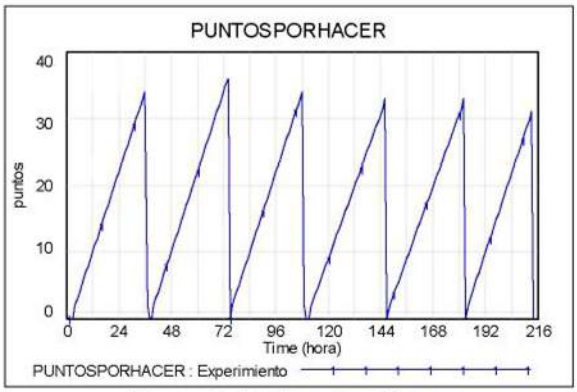


Figura 9–Puntos Por Hacer

Manteniendo el criterio inicial de que a cada punto de historia le corresponde una tarea, en la Figura 10 se observa el comportamiento de la variable TAREASPLANIFICADAS que representa el número de tareas según lo acordado inicialmente por el Team.

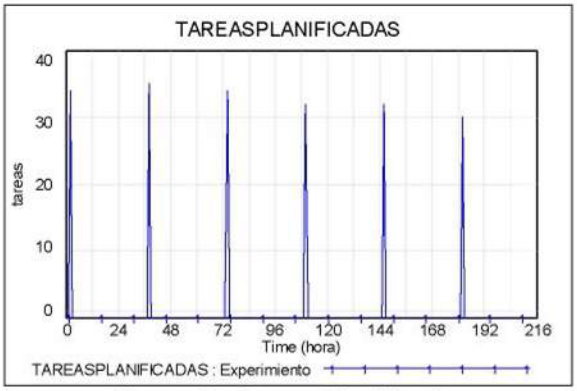


Figura 10 –Tareas Planificadas

Como se mencionó al inicio de este Experimento, y como se estimó en los parámetros iniciales de este caso se consideran las tareas que presenten errores de codificación y las tareas extras planificadas a desarrollar. Los valores que estas dos variables se presentan respectivamente en la Figuras 11y 12 respectivamente.

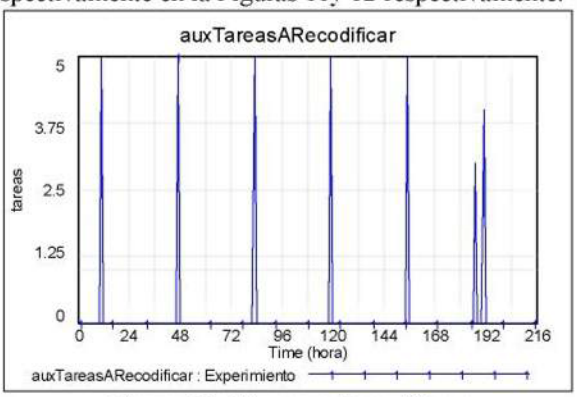


Figura 11 –Tareas a Recodificar

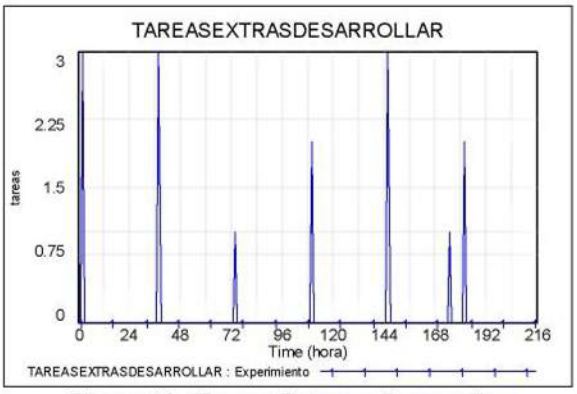


Figura 12 –Tareas Extras a Desarrollar

Dada las características particulares de este escenario dónde se presentan tareas extras planificadas y por errores de codificación (Figuras 11 y 12) se muestran en la Figura 13 de manera superpuesta las tareas originales, y las tareas originales a las que se adicionaron las tareas extras planificadas. Aunque es sutil la diferencia se

aprecia que en la corrida la cantidad de tareas es mayor que en la corrida donde no se incluían las tareas extras.

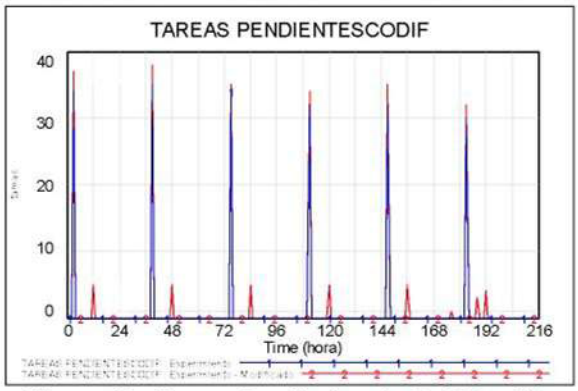


Figura 13 –Tareas Pendientes de Codificación

En la Figura 14 se observan de manera superpuesta las corridas donde se incluyeron las Tareas codificadas Con y Sin tareas extras, de manera similar a la Figura 13 en este caso se observa que al inicio de cada Sprint la corrida involucra mayor cantidad de tareas, y por lo tanto al final de cada Sprint quedan tareas pendientes por completar, dado que se mantuvo la velocidad estimada inicialmente.

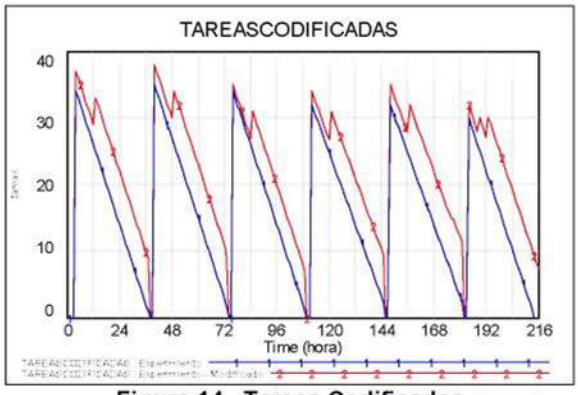


Figura 14 –Tareas Codificadas

En la Figura 15 se observa el comportamiento de la variable que representa a las pruebas que deberán ser realizadas sobre las diferentes tareas codificadas.

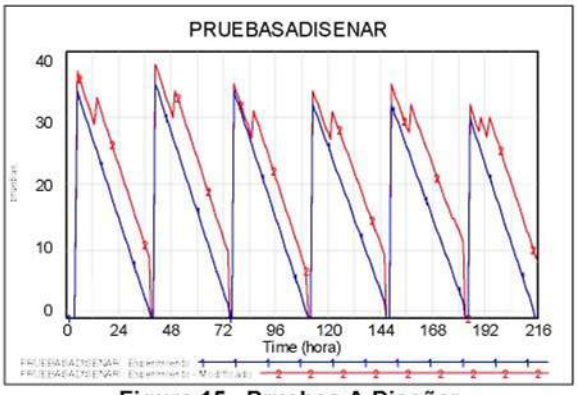


Figura 15– Pruebas A Diseñar

Como se mencionó anteriormente en este experimento, se consideran las tareas que por presión en el plazo pudieran sufrir el Team y repercutir en la codificación de las tareas. Es por ello que en la Figura 16 se observan de manera superpuesta el resultado de las diferentes pruebas realizadas sobre las tareas codificadas y aquellas tareas codificadas que no presentaron errores de codificación.

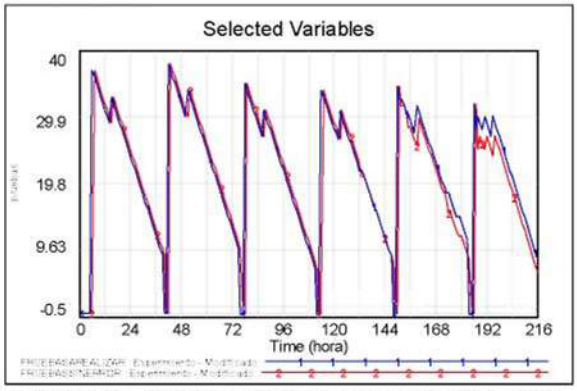


Figura 16– Pruebas Sin Errores de Codificación

En la siguiente Figura 17, se presentan las tareas que no presentaron errores de codificación o lógicos, junto a aquellas que si presentaron errores. Si bien es mínima la diferencia, en la figura correspondiente a las tareas sin errores se presentan pequeñas caídas en los momentos donde las tareas con error fueron detectadas.

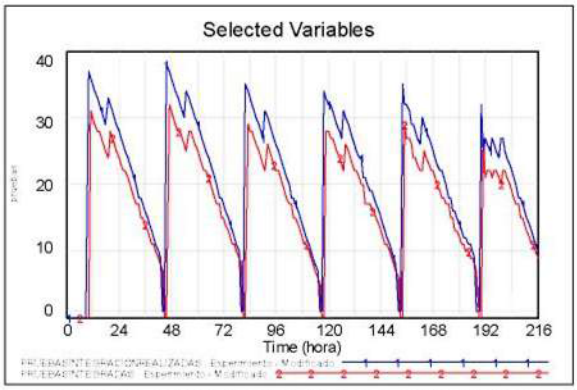


Figura 17– Pruebas Con Errores y Sin Errores

5.4. Mejoras Propuestas

En este apartado se presentan dos posibles mejoras frente a la situación que se presentó en el desarrollo del proyecto bajo sus condiciones estimadas inicialmente.

Una de las soluciones es el resultado de realizar una variación en la velocidad diaria de trabajo manteniendo los tiempos iniciales establecidos para la duración de cada Sprint. La otra consiste en la asignación de más horas a la duración de cada Sprint. Ambas soluciones se presentan de manera conjunta a fin de poder realizar una comparación de los resultados obtenidos. En ambos casos los resultados que se obtuvieron corresponden a las corridas que incluían tareas extras.

Las Figuras 18, 20, 22 y 24, son el resultado de corregir la situación planteada inicialmente con un ajuste de la velocidad de desarrollo de tareas, logrando de esta manera que todas las tareas planificadas y las tareas no planificadas finalicen en el tiempo estimado inicialmente.

Las Figuras 19, 21, 23 y 25, son el resultado de mantener la velocidad inicial, pero donde a los Sprints se le adicionaron más horas de trabajo

Se observa en la Figura 18 donde se realizó un ajuste de velocidad del 1.5%, quedan pequeños espacios de tiempo sin actividad entre los Sprints dado que el ajuste fue similar en todos los Sprints.

En la Figura 19 se destaca el aumento en la duración del proyecto cuando son adicionadas horas de trabajo a los Sprints, lo que derivaría en un retraso del proyecto y en un aumento en el cansancio del Team.

Ambas figuras presentan además el grafico de BurdownChart a modo de comparación entre lo que se estimó inicialmente y la manera en la que se desarrollaron las tareas planificadas más las tareas extras. Es válido aclarar que si bien aparecen unidades diferentes el análisis y la comparación son válidos ya que a cada punto de historia le corresponde una tarea.

Así en la figura 18 el desarrollo estimado inicialmente es similar al desarrollo real a lo largo del proyecto. Por su parte en la figura 19 es notoria la diferencia existente entre lo planificado y lo realizado. Esta última situación llevaría al fracaso del proyecto.

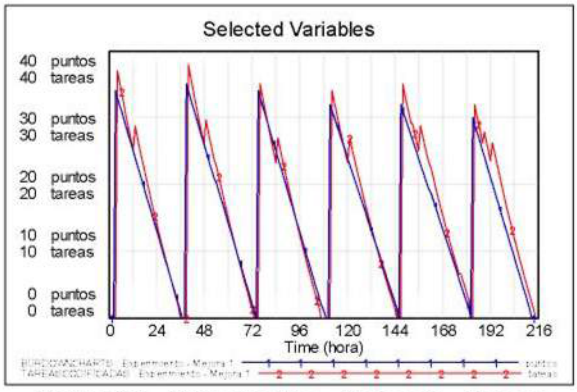


Figura 18 – Experimento – Mejora 1 – Tareas Codificadas

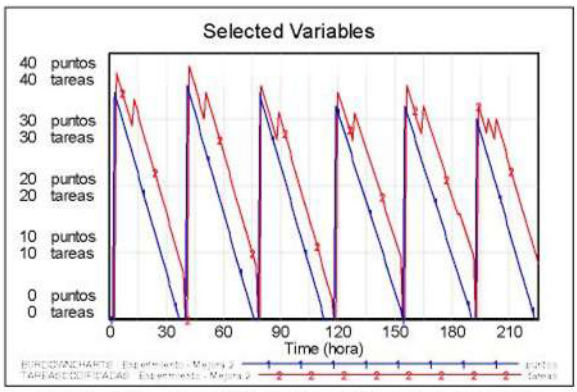


Figura 19 – Experimento – Mejora 2 –Tareas Codificadas

Al igual que en el caso de las Tareas Codificadas, las Pruebas a Diseñar también logran completarse en su totalidad en la primera de las mejoras propuestas Pudiendo observarse esto en la Figura 20.

En la 21 se presenta el resultado de adicionar más horas a los Sprints del Experimento haciéndose evidente que no se alcanzaron a realizar todas las pruebas planificadas.

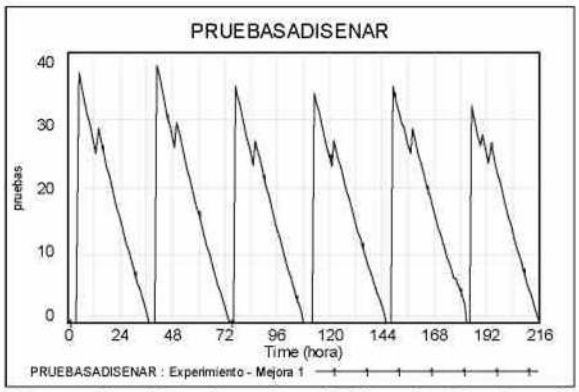


Figura 20 – Experimento – Mejora 1 – Pruebas A Diseñar

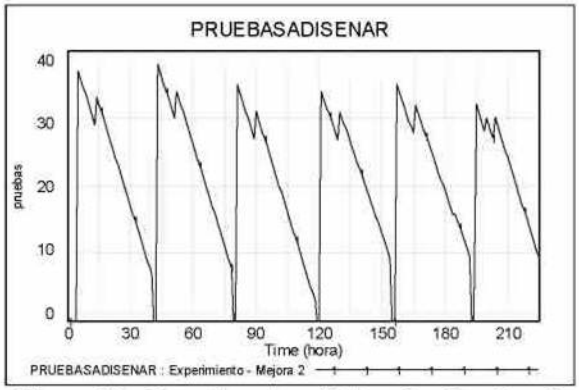


Figura 21 – Experimento – Mejora 2 – Pruebas A Diseñar

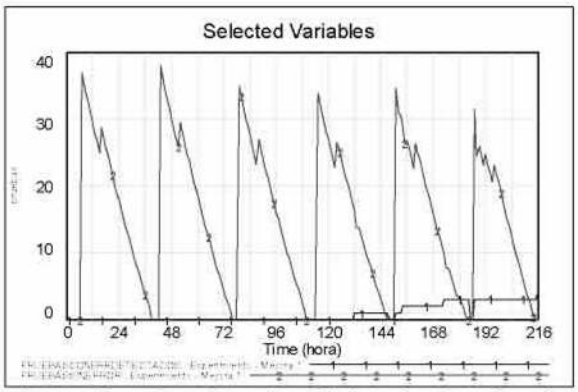


Figura 22 – Experimento – Mejora 1 – Pruebas de Codificación

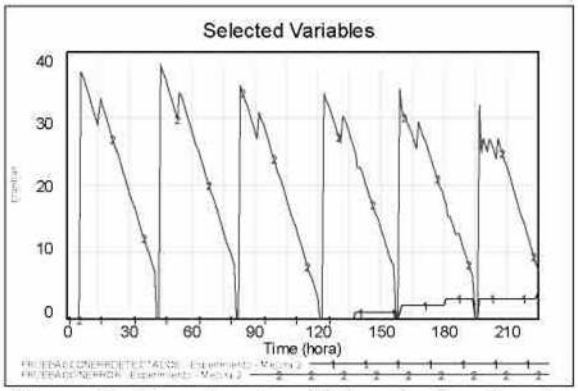


Figura 23– Experimento – Mejora 2 – Pruebas de Codificación

En las Figuras 22 y 23 se presentan los resultados de las pruebas realizadas y las pruebas que resultaron exitosas. tanto para la mejora donde se realizó un ajuste de velocidad como. en el caso donde se adicionaron horas a los Sprints respectivamente.

Como ocurrió tanto en las Pruebas a Diseñar como en la Codificación de Tareas. en las Pruebas de Integración Realizadas en la primera de las mejoras propuestas se logra completar la totalidad de lo estimado inicialmente. En cambio las pruebas de integración para la segunda mejora propuesta no llegan cumplirse en su totalidad.

En las Figuras 24 y 25 se presentan los resultados de las pruebas de integración realizadas y las pruebas que resultaron exitosas. tanto para la mejora donde se realizó un ajuste de velocidad como. en el caso donde se adicionaron horas a los Sprints respectivamente.

Puede observarse en este caso como para la propuesta de mejora donde se adicionaron horas a los diferentes Sprints se producen espacios entre la finalización y el inicio de un Sprint. dada esta situación podría decirse que el número de horas adicionados fue mayor al necesario. acarreando esto una demora en la entrega del producto final.

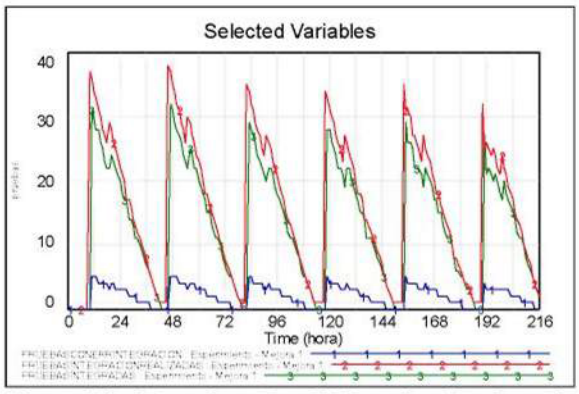


Figura 24 – Experimento – Mejora 1 – Pruebas de Integración

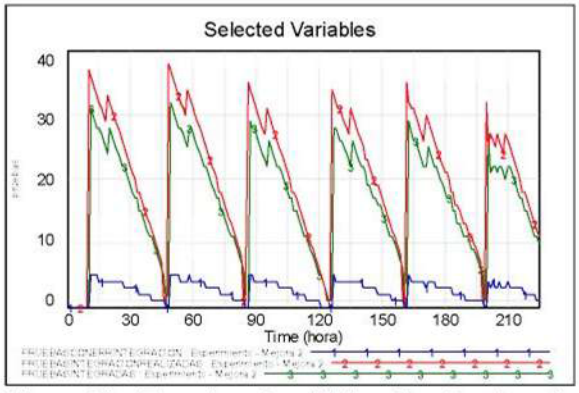


Figura 25 – Experimento – Mejora 2 – Pruebas de Integración

5.5. Conclusión del experimento.

Como se puede apreciar en este experimento la cantidad excesiva de errores y las ausencias de los integrantes del Team hicieron que el proyecto bajo las condiciones iniciales presentara un alto número de tareas pendientes de completar al final de los diferentes Sprints. por lo que bajo esas circunstancias el proyecto fracasaría inevitablemente.

Debido a esto el Scrum Master podría detectar esta situación de modo tal que en el Sprint Planning las plantearía y propondría alguna otra política de gestión del proyecto o bien optar por alguna mejora. En este experimento se optó por un ajuste a la velocidad de trabajo diario y la adición de más horas a los Sprints.

La propuesta de mejora mediante la adición de más horas a los Sprints y en consecuencia al proyecto. no resulta una opción viable ya que aun después de adicionar un promedio de 2 horas a cada Sprint el número de tareas pendientes de desarrollar es elevado. Aun. si se agregaran más horas el problema solamente se extendería ya que la mayor adición de horas traería aparejado una mayor cantidad de errores por cansancio y un atraso mayor en la finalización del proyecto.

Por otro lado la propuesta donde se realizó un ajuste de la velocidad de trabajo diario resulto la opción más viable dado que se pudieron llevar a cabo todas las tareas planificadas y las diferentes pruebas de control de errores.

6. Conclusiones

En este trabajo se presentó un caso de validación y un experimento realizado con un "Modelo Dinámico de Simulación para Proyectos de Software que utilizan Scrum". Este modelo puede ser de utilidad para los Scrum Master y el Team analizar el efecto del uso conjunto de la metodología Scrum. Bloques de Tiempo. Artefactos y Reglas en la gestión de los mismos en diferentes escenarios. La flexibilidad de modelo permite modificar los valores de los parámetros tanto al inicio de la simulación como al momento de la ejecución de la misma. Dentro de los parámetros que se pueden establecer previo al inicio de cada simulación se encuentran: la duración y la velocidad de cada Sprint. la velocidad estimada de desarrollo de las tareas. Factores de Cansancio. de Presión en el plazo. Cantidad de integrantes del Team según su experiencia en la metodología y las tareas extras que se prevén puedan surgir. A través de la modificación de valores de los parámetros durante su ejecución el usuario puede establecer o modificar la cantidad de integrantes del Team que abandonan el proyecto. clasificar al Team mediante la asociación de estos a su experiencia en Scrum en Juniors o Expertos. cambiar la cantidad de horas estimadas de duración del proyecto. generar horas extras e inasistencia de los integrantes de manera determinística o pseudoaleatoria. entre otros.

Finalmente. luego de validar y experimentar con el modelo. se ha llegado a la conclusión de que el mismo puede ser utilizado como herramienta para evaluar el impacto de políticas alternativas de gestión y la detección de cuellos de botella.

7. Trabajos Futuros

Como trabajos futuros se planea Adicionar otros sub-sistemas que permitan ampliar el ámbito del modelo. lo que permitiría una herramienta que facilite aún más las tareas previas al inicio del proyecto al Scrum Master y al Team. Dentro de estos sub-sistemas podrían nombrarse: cálculo de costos. comunicación en el Team. Intercambio de Roles. Adquisición de Experiencia. por nombrar solamente algunos. Así también. se pretende avanzar con la construcción de modelos similares para otras metodologías consideradas ágiles como Cristal Clear y Crystal Orange [15][16] o incluir prácticas como por ejemplo Test DrivenDevelopment[17].

Por otra parte se planea utilizar este simulador en cátedras relacionadas con la Ingeniería de Software, con el fin de entrenar a futuros Scrum Masters y Teams.

Finalmente, es preciso construir bases de datos de Proyectos Scrum que contengan datos de proyectos de software reales llevados a cabo, ya que actualmente resulta difícil contar con datos post mortem de proyectos gestionados con métodos ágiles.

Bibliografía

[1] Ken y Sutherland. Jeff. Schwaber, *Agile Software Development with Scrum*. Primera ed.: Prentice Hall, 2001.

[2] Kim E. Van. Kishore Sengupta. and Luk N. Van., "Dynamics of Agile Software Development." 2009.

[3] Xiaoying Konga, Li Liu, and Chen Jing, "Modeling Agile Software Maintenance Process Using Analytical Theory of Project Investment," *International Conference on Advances in Engineering 2011*. 2011.

[4] Tamara Kasiak and Diego Alberto Godoy, "Simulación de Proyectos de Software desarrollados con XP," *XIV Workshop de Investigadores en Ciencias de la Computación.*, 2012.

[5] Diego Alberto Godoy and Kasiak Tamara, "Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP," in *Actas XVIII Congreso Argentino de Ciencias de la Computación*, 2012, p. 10.

[6] Firas Glaiel, "Agile Project Dynamics: A Strategic Project Management Approach to the Study of Large-Scale Software Development Using System Dynamics," Massachusetts Institute of Technology , Tésis de Máster 2012.

[7] Diego Alberto Godoy, Edgardo A. Belloni, Henry Kotynski, Hector H Dos Santos, and Eduardo Omar Sosa, "Simulando Proyectos de Desarrollo deSoftware Administrados con Scrum," in *XVI Workshop de Investigadores en Ciencias de la Computación RedUNCI*, Ushuaia, 2014.

[8] J Aracil, *Dinámica de Sistemas*. Madrid, España: Alianza Editorial, 1997.

[9] Torrealdea J, *Dinámica de Sistemas. Elementos y Estructuras de un Modelo.Construyendo modelos.*: Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco, 2003.

[10] Ventana System Inc. (2013) Vensim. [Online]. http://www.vensim.com

[11] MaríaLaura. Citón. "Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad," Universidad de Mendoza - Facultad de Ingeniería. Mendoza, Trabajo Final de Carrera Ingeniería en Informática 2006.

[12] Tiago Keller Ferreira. "Aplicação Do Processo Ágil De Gerenciamento Scrum No Desenvolvimento De Um Jogo Digital.," UFSM, Informática/UFSM - Biblioteca Digital de Trabalhos de Graduação. , Estudo De Caso Em Empresa 2008.

[13] V. A Mahnic, "case study on agile estimating and planning using scrum," *Electronics & Electrical Engineering*, vol. 111, no. 5, 2011.

[14] Héctor MudarraTeruel, "Automatización de sistemas de desarrollo ágil Scrum: Team & Role -," sitio Web temoa : Portal de Recursos Educativos Abiertos (REA), Memoria delProyecto de Fin de Carrera de Ingeniería Informática Bellaterra. Junio de 2010 2010.

[15] Alistair Cockburn. *Crystal Clear, A Human-Powered Methodology for Small Teams.*: Addison-Wesley Professional, 2004.

[16] Cockburn, Alistair. *Agile Software Development: The Cooperative Game.*: Addison-Wesley Professional, 2006.

[17] L Madeyski. *Test-Driven Development - An Empirical Evaluation of Agile Practice.*: Springer, 2010.