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 [9], 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 [10] 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

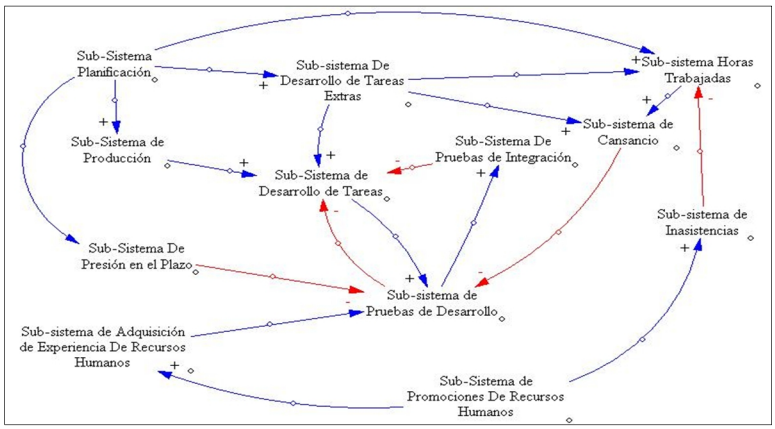

Figura1-Diagrama de relaciones causales entre Sub-sistemas.

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.

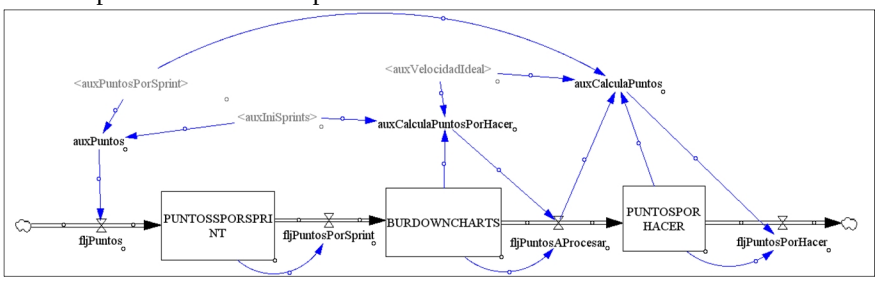

Figura2 - Diagrama de Forrester Sub-sistema de Producción

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 Scrum 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.

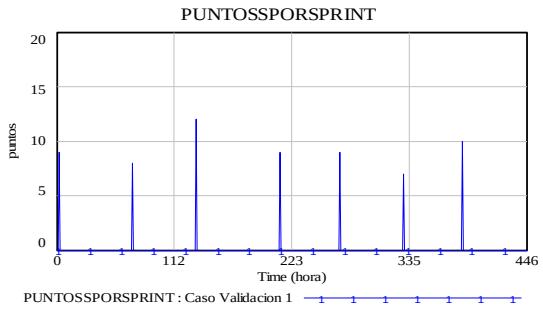

Figura 3 - Puntos Planificados Por Sprint

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.

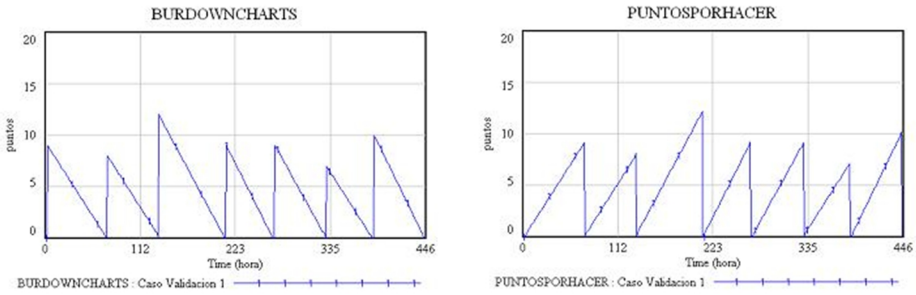

Figura 4 - Desarrollo ideal de los puntos y Puntos por hacer

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.

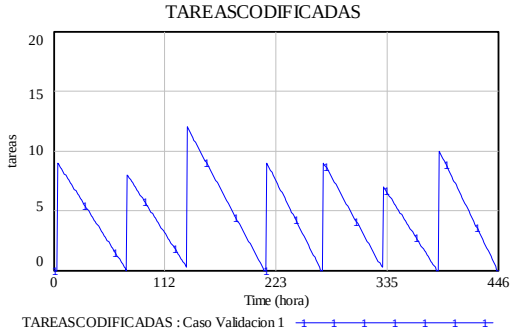

Figura 5 - Tareas Codificadas Por Sprint.

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).

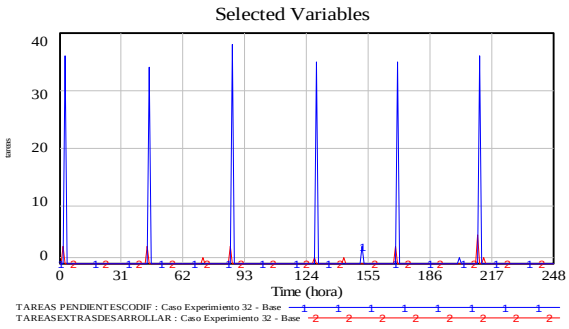

Figura 6 – 1) Tareas Extras Planificadas. 2) Tareas Extras No Planificadas

En la Figura 7 se observan superpuestas las corridas donde se consideró el proyecto sin tareas extras y con tareas extras.

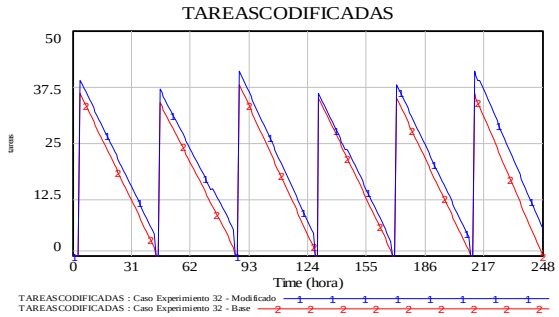

Figura 7 - 1) Proyecto sin tareas extras. 2) Proyecto con Tareas Extras

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.

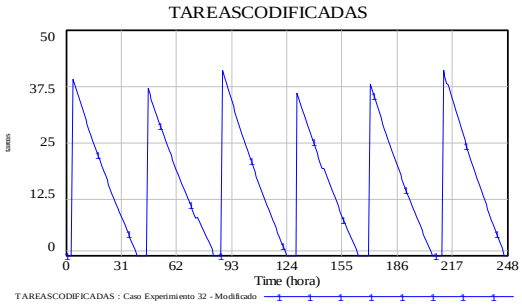

Figura 8– Sprints luego de ajustar la velocidad del desarrollo

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.

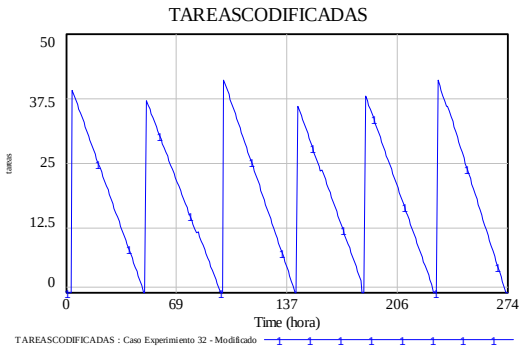

Figura 9–Sprints luego de aumentar en 10 horas la duración del proyecto.

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 lScrum 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.