

UNIVERSIDAD NACIONAL DEL CENTRO DEL PERU



FACULTAD DE INGENIERÍA DE SISTEMAS

TESIS

**APLICACIÓN DE LA METODOLOGÍA SCRUM PARA
INCREMENTAR LA PRODUCTIVIDAD DEL PROCESO
DE DESARROLLO DE SOFTWARE EN LA EMPRESA
CCJ S.A.C. LIMA**

PRESENTADA POR:

MALPICA VELÁSQUEZ, Carlos Jesús

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

**HUANCAYO – PERÚ
2014**

ASESOR:

Mg. Richard Yuri Mercado Rivas

AGRADECIMIENTOS:

Deseo expresar muestras de agradecimiento:

A DIOS

Por darme la vida y enseñarme a valorar cada minuto.

A MIS PADRES

Por darme el ejemplo y la fuerza para seguir adelante.

A MI ALMA MATER

Por albergarme durante mi formación como ingeniero.

A MI ASESOR

Por su guía y seguimiento en el desarrollo de la tesis.

A MIS MAESTROS DE LA FIS

Por brindarme sus conocimientos y experiencias.

A LA EMPRESA CCJ S.A.C.

Por su apoyo e interés para el éxito del presente trabajo.

DEDICATORIA

A mis padres Edith y Carlos por ser ejemplo de lucha constante y de unidad en los momentos difíciles.

RESUMEN

La presente tesis intitulada “Aplicación de la Metodología Scrum para incrementar la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C. Lima”, se ha enfocado en la Unidad de Negocio de Tecnologías de Información de CCJ, una corporación con la visión de servicio en el diseño, desarrollo y ejecución de proyectos en ingeniería. Sin embargo, el modelo con el que viene desarrollando los proyectos de software era adaptado del Ciclo de Vida en Cascada, el cual es inadecuado por su carencia de agilidad y flexibilidad, generando desfases en tiempos y costos.

Bajo este contexto, la Metodología Scrum se presenta como una atractiva posibilidad debido a su naturaleza ágil, lo cual implica un carácter adaptable, orientado a las personas más que a los procesos y que emplea la estructura de desarrollo ágil. A diferencia del Ciclo de Vida en Cascada, posee agilidad, flexibilidad permitiendo el incremento de la calidad y la reducción notable de tiempo y costos.

El modelo aplicativo Scrum consta de cinco fases: Definición del backlog del producto, Planificación del sprint, Scrum diario, Revisión del sprint y Retrospectiva del sprint. Para su correcta aplicación se comenzó con la visión general del producto, proporcionado por el product owner, ésta información se estructura en el backlog del producto, que contiene los sprint backlogs que son especificaciones funcionales de las partes con mayor prioridad de desarrollo. Estos sprints fueron planificados uno a uno y se llevaron a cabo en un periodo de 1 a 4 semanas mediante reuniones diarias donde participaron el scrum master y el product owner. Cada periodo de desarrollo atravesó por la revisión del sprint, y al existir ciertas variaciones respecto al requerimiento inicial se dio la retrospectiva del sprint. Todo el proceso finalizó con la producción de un incremento operativo del producto validado por el cliente, lo que Scrum denomina “potentially shippable”.

Con la aplicación de Scrum, se obtuvo como resultado: la reducción del número de días de retraso a cero días, pudiéndose cumplir con los entregables del proyecto en el plazo establecido, no incurriendo en costos adicionales. Además se logró mejorar el clima laboral con las reuniones diarias que establece Scrum.

Concluyendo, se plantea que Scrum al ser un proceso de desarrollo iterativo e incremental se puede usar para cualquier área de una empresa como Tecnologías de Información, Administración, Operaciones, entre otras, ya que sirve para planificar, ordenar, reportar el trabajo del día a día, semanal, mensual, anual. Por lo que, se recomienda implementar Scrum en su empresa, porque permite la creación de equipos auto-organizados impulsando la participación activa de todos los miembros del equipo, y la comunicación verbal entre todos los integrantes y disciplinas involucradas en el proyecto.

ABSTRACT

The present thesis entitled "Application of the Scrum Methodology for increasing the productivity of software development process in the CCJ S.A.C. Business Lima" has focused on the Business Unit Information Technologies of CCJ, a corporation with the vision of service in the design, development and implementation of engineering projects. However, the model that has been developing software projects was adapted of the Lifecycle Cascade, which is inappropriate for its lack of agility and flexibility, resulting in time and cost offsets.

In this context, the Scrum Methodology is presented as an attractive option owing to its agile nature, which implies a customizable character, people-oriented rather than process and employees an agile development structure. Unlike Lifecycle Cascade, has agility, flexibility allowing increased quality and significant reduction of time and costs.

The Scrum application model consists of five phases: Define the product backlog, Sprint planning, Daily scrum, Sprint review and Sprint retrospective. For proper application is started with overview of the product, provided by the product owner, this information is structured in the product backlog that contains sprint backlogs that are functional specifications of the parts with more development priority. These sprints were planned one by one and were carried out in a shorter period to 30 days by daily meetings where the scrum master and product owner participated. Each period of development went through the sprint review, and to be certain variations from the original requirement sprint retrospective was given. The whole process ended with the production of an operational increase of product validated by the customer, it Scrum calls "potentially shippable".

With the application of Scrum, was obtained as result: reducing the number of days of delay to zero days, being able to achieve the deliverable by the deadline, incurring no additional cost. In addition it was possible to improve the working environment with daily Scrum meetings.

In conclusion, we propose that Scrum to be an iterative and incremental development can be used for any area of a company like Information Technology , Management, Operations, among others, as it serves to plan, order, report to the day job daily , weekly , monthly, yearly. So, it is recommended to implement Scrum in your business, because it allows the creation of self-organizing teams and encourage the active participation of all team members, and verbal communication between all members and disciplines involved in the project.

ÍNDICE

	Pág.
ASESOR	ii
AGRADECIMIENTOS	iii
DEDICATORIA	iv
RESUMEN	v
ABSTRACT	vi
ÍNDICE	vii
ÍNDICE DE GRÁFICOS Y TABLAS	ix
INTRODUCCIÓN.....	1
CAPÍTULO I	
GENERALIDADES	
1.1. PLANTEAMIENTO DEL PROBLEMA	3
1.1.1. LA INDUSTRIA DE SOFTWARE EN EL MUNDO.....	3
1.1.2. LA INDUSTRIA DE SOFTWARE EN EL PERÚ.....	7
1.1.3. UNIDAD DE TECNOLOGÍAS DE INFORMACIÓN DE LA EMPRESA CCJ.....	13
1.1.3.1. PRINCIPALES CLIENTES DE LA UNIDAD DE NEGOCIO DE TI	14
1.1.3.2. METODOLOGÍA DE DESARROLLO EMPLEADA EN LA UNIDAD DE NEGOCIO DE TI	15
1.1.3.3. INDICADORES DE PRODUCTIVIDAD EN EL PROCESO DESARROLLO DE SOFTWARE	17
1.2. FORMULACIÓN DEL PROBLEMA.....	27
1.3. OBJETIVOS.....	27
1.4. JUSTIFICACIÓN	27
1.4.1. JUSTIFICACIÓN TEÓRICA	27
1.4.2. JUSTIFICACIÓN PRÁCTICA	27
1.4.3. JUSTIFICACIÓN METODOLÓGICA	28
1.5. HIPÓTESIS	28
1.5.1. HIPÓTESIS GENERAL.....	28
1.5.2. OPERACIONALIZACIÓN DE LAS VARIABLES E INDICADORES DE LA HIPÓTESIS	28
1.6. DISEÑO METODOLÓGICO.....	29
1.6.1. TIPO DE INVESTIGACIÓN.....	29
1.6.2. NIVEL DE INVESTIGACIÓN.....	29
1.6.3. SISTEMA DE REFERENCIA.....	29
CAPÍTULO II	
MARCO DE REFERENCIA	
2.1. ANTECEDENTES	30
A.1. Estudio de la Aplicación de Metodologías Ágiles para la Evolución de productos software	30
A.2. Mejora del Proceso de Software de una Empresa Desarrolladora de Software: Caso COMPETISOFT – PERÚ DELTA	32
A.3. Automatización de Sistemas de Desarrollo Ágil – SCRUM: TEAM & ROLE	33
A.4. Adaptation of a software development methodology incorporating Social Network Analysis and Situational Leadership.....	34
A.5. Usabilidad en el proceso de desarrollo de SCRUM	35
2.2. MARCO TEÓRICO.....	37
2.2.1. INTRODUCCIÓN AL MODELO ÁGIL DE DESARROLLO DE SOFTWARE	37
2.2.2. CÓMO SER ÁGIL. ALGUNAS PRÁCTICAS.....	43
2.2.3. REVISIÓN DE OTRAS METODOLOGÍAS ÁGILES	46
2.2.4. ORIGEN DE LA METODOLOGÍA SCRUM.....	49
2.2.5. ¿POR QUÉ APLICAR SCRUM?	49
2.2.6. OBJETIVOS Y PREFERENCIAS DE LA GESTIÓN ÁGIL.....	51
2.2.7. CONTROL DE LA EVOLUCIÓN DEL PROYECTO	53
2.2.8. VISIÓN GENERAL DEL PROCESO.....	54
2.3. MODELO APLICATIVO.....	57
2.4. MARCO CONCEPTUAL.....	82

CAPÍTULO III

INTERVENCIÓN METODOLÓGICA

3.1.	FASE N° 1: DEFINICIÓN DEL BACKLOG DEL PRODUCTO	85
3.2.	FASE N° 2: PLANIFICACIÓN DEL SPRINT	90
3.3.	FASE N° 3: SCRUM DIARIO	96
3.4.	FASE N° 4: REVISIÓN DEL SPRINT	99
3.5.	FASE N° 5: RETROSPECTIVA DEL SPRINT	105

CAPÍTULO IV

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

4.1.	PRESENTACIÓN DE RESULTADOS	109
4.2.	DISCUSIÓN DE RESULTADOS	114
4.3.	VALIDACIÓN DE HIPÓTESIS	116

CONCLUSIONES

RECOMENDACIONES

REFERENCIAS

ANEXOS

ÍNDICE DE GRÁFICOS Y TABLAS

	Pág.
Gráfico N° 1. 1 Tasa de penetración de Internet por zonas geográficas	5
Gráfico N° 1. 2 Ventas Totales	10
Gráfico N° 1. 3 Ventas Nacionales y al Exterior de Software Peruano	11
Gráfico N° 1. 4 Principales Destinos al Exterior 2010.....	11
Gráfico N° 1. 5 Mercado Mundial 2010: 625 billones de euros	12
Gráfico N° 1. 6 Principales clientes de CCJ – Unidad de negocio de TI	14
Gráfico N° 1. 7 Flujo de la Metodología de software empleada por CCJ – TI	15
Gráfico N° 1. 8 Distribución del riesgo en un desarrollo en cascada.....	16
Gráfico N° 1. 9 Distribución del riesgo en un desarrollo ágil.....	16
Gráfico N° 1. 10 Dos culturas, Desarrollo Convencional vs. Scrum	17
Gráfico N° 1. 11 Estado de los Proyectos de Software desarrollados en CCJ S.A.C. Enero – Diciembre de 2013.....	21
Gráfico N° 1. 12 Cronograma del proyecto “La Positiva – Modificación Sistema de Gestión de Cobranzas - Antamina”	23
Gráfico N° 1. 13 Gráfico lineal de Tiempo Asignado vs. Tiempo Utilizado en los.....	24
 Gráfico N° 2. 1 Manifiesto para el desarrollo ágil de software	 41
Gráfico N° 2. 2 Desarrollo tradicional vs. Desarrollo ágil	50
Gráfico N° 2. 3 Visión general del proceso	54
Gráfico N° 2. 4 Distribución clásica de los roles para Scrum.....	56
Gráfico N° 2. 5 Modelo aplicativo Scrum	58
Gráfico N° 2. 6 Ejemplo de Product Backlog o Pila de Producto	60
Gráfico N° 2. 7 Variables involucradas en la definición de una historia de usuario	63
Gráfico N° 2. 8 Ejemplo típico de una agenda de reunión Scrum	65
Gráfico N° 2. 9 Esquema de Pila de producto.....	66
Gráfico N° 2. 10 Ejemplo de velocidad estimada al principio y final de un Sprint	67
Gráfico N° 2. 11 Ejemplo de comienzo de un Sprint	69
Gráfico N° 2. 12 Ejemplo de división de una historia en tareas	71
Gráfico N° 2. 13 Ejemplo de subdivisión de una tarea	71
Gráfico N° 2. 14 Ejemplo de página de información de Sprint.....	73
Gráfico N° 2. 15 Estructura de una Tabla de tareas.....	74
Gráfico N° 2. 16 Funcionamiento de una Tabla de tareas.....	74
Gráfico N° 2. 17 Tablón de tareas después del primer Scrum.....	75
Gráfico N° 2. 18 Tabla de tareas unos días más tarde.....	76
Gráfico N° 2. 19 Tablón de tareas después del primer Scrum.....	76
Gráfico N° 2. 20 Ejemplo de pizarra de una reunión de retrospectiva Scrum.....	80
 Gráfico N° 3. 1 Interfaz del Cotizador vehicular actual en Excel – La Positiva SegurosS.A.....	 86
Gráfico N° 3. 2 Cronograma del proyecto “Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.”.....	87
Gráfico N° 3. 3 Esquema general de la arquitectura del proyecto	88
Gráfico N° 3. 4 Base de datos del cliente del proyecto Inrule Technology	90
Gráfico N° 3. 5 Sprint backlog para el proyecto Inrule Technology	97
Gráfico N° 3. 6 Diagrama burndown para el Sprint backlog 1.....	97
Gráfico N° 3. 7 Diagrama burndown para el Sprint backlog 2.....	98
Gráfico N° 3. 8 Diagrama burndown para el Sprint backlog 3.....	98
Gráfico N° 3. 9 Modelo de datos de la base de datos InruleDB conteniendo un error.....	99
Gráfico N° 3. 10 Creación de Entidad en Inrule definidas en el Modelo de datos	100
Gráfico N° 3. 11 Creación del Lenguaje de Reglas en Inrule según la lógica de negocio de La Positiva Seguros S.A.....	101
Gráfico N° 3. 12 Creación de los campos que lanzarán el resultado de la aplicación de la regla	101
Gráfico N° 3. 13 Carga de la data en Inrule a través de Inline Table	102
Gráfico N° 3. 14 Web Service en Visual Studio .Net 2012.....	103
Gráfico N° 3. 15 Aplicativo en Visual Studio .Net 2012	104
Gráfico N° 3. 16 Aplicativo en Visual Basic 6.0	104
Gráfico N° 3. 17 Código fuente de Visual Studio .Net 2012.....	105
Gráfico N° 3. 18 Modelo de datos de la base de datos InruleDB con levantamiento del error	105

Gráfico N° 4. 1 Esquema del flujo descriptivo del proyecto	110
Gráfico N° 4. 2 Parámetros de entrada en Inrule Technology	110
Gráfico N° 4. 3 Parámetros de salida en Inrule Technology	111
Gráfico N° 4. 4 Ingreso de parámetros - Software de Cotización de Seguro Vehicular .Net.....	111
Gráfico N° 4. 5 Ingreso de parámetros - Software de Cotización de Seguro Vehicular VB6.0.....	112
Gráfico N° 4. 6 Ingreso de parámetros – Servicio Web del Cotizador de Seguro Vehicular	113
Gráfico N° 4. 7 Resultados obtenidos - Software de Cotización de Seguro Vehicular .Net	114
Gráfico N° 4. 8 Resultados obtenidos - Software de Cotización de Seguro Vehicular VB6.0	115
Gráfico N° 4. 9 Resultados obtenidos – Servicio Web del Cotizador de Seguro Vehicular	115
Gráfico N° 4. 10 Gráfico lineal de la tendencia del proceso de desarrollo ágil Scrum – Proyecto.....	118
Tabla N° 1. 1 Disponibilidad de las nuevas tecnologías 2009-2010.....	4
Tabla N° 1. 2 América Latina. Participación de los gastos realizados en cada país en el gasto total en el conjunto de países.....	6
Tabla N° 1. 3 ¿A qué se dedica la empresa?.....	8
Tabla N° 1. 4 Uso de plataformas tecnológicas y lenguajes de programación	9
Tabla N° 1. 5 Gestores de Base de Datos más utilizados	9
Tabla N° 1. 6 Salario promedio mensual en dólares - 2011	10
Tabla N° 1. 7 Relación entre los factores de calidad del software y las métricas	19
Tabla N° 1. 8 Indicadores de productividad por desarrollador empleado en CCJ S.A.C.	20
Tabla N° 1. 9 Lista de Proyectos de Software que debían ser ejecutados en CCJ S.A.C.	21
Tabla N° 1. 10 Duración, Retraso y Estado de los Proyectos de Software en CCJ S.A.C.....	22
Tabla N° 1. 11 Cuadro de Tiempo Asignado vs. Tiempo Utilizado por Desarrollador.....	22
Tabla N° 1. 12 Cuadro que refleja la insatisfacción de los trabajadores	25
Tabla N° 1. 13 Cuadro que refleja la insatisfacción de los clientes	25
Tabla N° 1. 14 Razones por las que no cambiarían la metodología de desarrollo	26
Tabla N° 3. 1 Product Backlog del proyecto proporcionado por el dueño del producto	89
Tabla N° 3. 2 Product Backlog del proyecto conteniendo la Importancia y la Estimación inicial	91
Tabla N° 3. 3 Sprints backlog definidos en la primera reunión de planificación	93
Tabla N° 3. 4 Estado final de los tareas del Product Backlog del proyecto	106
Tabla N° 3. 5 Cierre del proyecto con Scrum.....	107
Tabla N° 4. 1 Estadística de ejecución de Inrule Technology	113
Tabla N° 4. 2 Indicadores de productividad ANTES por desarrollador empleado en CCJ S.A.C.....	116
Tabla N° 4. 3 Indicadores de productividad DESPUÉS por desarrollador empleado en CCJ S.A.C..	117
Tabla N° 4. 4 Cuadro de Tiempo Asignado vs. Tiempo Utilizado por Desarrollador.....	117
Tabla N° 4. 5 Cuadro que refleja la satisfacción de los trabajadores aplicando Scrum	118
Tabla N° 4. 6 Cuadro que refleja la satisfacción de los clientes aplicando Scrum.....	119

INTRODUCCIÓN

La industria de las tecnologías de la información a nivel mundial se encuentra fuertemente sujeta al desarrollo económico de cada nación, y por consiguiente la industria del software se encuentra bajo las mismas condiciones. Mientras que, la industria de software en el Perú, conformada por empresas que ofrecen servicios de desarrollo de tecnologías de información, es cada vez más creciente. Cabe resaltar que la industria del software peruana es mayoritariamente nacional ya que produce un 90% para el mercado local y sólo destina el 10% restante hacia el exterior, esto según las últimas estadísticas de APESOPT para el 2012. A pesar de esto, las exportaciones en el 2010 tuvieron un crecimiento de más del 25% respecto a las exportaciones del año anterior. CCJ es una empresa que contribuye a la producción de software para el mercado local en la ciudad de Lima, y cuenta con aproximadamente nueve años de operaciones en el sector.

La presente tesis “Aplicación de la Metodología Scrum para incrementar la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C. Lima”, está orientada a hacer más ágiles y eficientes los procesos de desarrollo en la Unidad de Negocio de Tecnologías de Información de la empresa CCJ. La estructura de la tesis consta de cuatro capítulos:

El capítulo I denominado Generalidades contempla el Planteamiento del Problema que describe la realidad actual y se enfoca en la situación problemática. La información mostrada presenta aspectos generales relacionados con la industria del software a nivel mundial, nacional y de la unidad de negocio de TI, evidenciando la naturaleza y magnitud del problema en estudio. Luego se realiza la Formulación del Problema, se incluye el Objetivo que se persigue a través del siguiente trabajo, la Justificación, la Hipótesis y el Diseño Metodológico, los cuales guiarán la investigación.

El capítulo II denominado Marco de Referencia muestra los Antecedentes que permiten evidenciar cómo se aplicó la metodología Scrum a otras organizaciones, el Marco Teórico enfocado en la metodología ágil Scrum, así como el Modelo Aplicativo el mismo que muestra la secuencia metodológica con la cual se pretende resolver el problema siguiendo las cinco fases propuestas: Definición del backlog del producto, Planificación del sprint, Scrum diario, Revisión del sprint y Retrospectiva del sprint. Culmina este capítulo con el Marco Conceptual correspondiente.

En el capítulo III denominado Intervención Metodológica se aplica la metodología Scrum, de acuerdo al modelo aplicativo del capítulo anterior, a un caso de estudio denominado: “Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.”. Y como las iteraciones son la base del desarrollo ágil, Scrum

gestiona su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.

En el cuarto y último capítulo denominado Análisis y Discusión de Resultados, se analizan los resultados obtenidos, los cuales son producto de la experimentación. Para finalmente poder validar la hipótesis general de la influencia de la metodología Scrum sobre el incremento de la productividad del proceso de desarrollo de software en la empresa CCJ.

Se finaliza con las conclusiones y recomendaciones, siendo la conclusión más importante que la productividad del proceso de desarrollo de software se ve influenciada de manera positiva por la aplicación de la metodología ágil Scrum en términos de reducción de tiempos y costos, logrando que los proyectos se realicen en los plazos estimados, existiendo un desfase de cero días y una pérdida neta de cero nuevos soles. Por lo que, se recomienda implementar Scrum en su empresa porque permite el incremento de la productividad de manera sostenible sobre todo para organizaciones que requieren rapidez de desarrollo en sus proyectos, agilidad en la migración de una tecnología a otra y poseen alta rotación de personal.

C. J. Malpica V.

CAPÍTULO I

GENERALIDADES

En el presente capítulo se abordarán aspectos referidos a la problemática la cual impulsó el desarrollo de la tesis, para ello se muestra información estadística sobre la industria de software en el mundo, en el Perú y la producción de software en la Empresa CCJ. También se detalla el objetivo que se desea alcanzar, la justificación del trabajo, la hipótesis que se desea verificar y el diseño metodológico, a fin de contribuir con el incremento de la productividad del proceso de desarrollo de software en CCJ.

1.1. PLANTEAMIENTO DEL PROBLEMA

Para contextualizar el planteamiento del problema desde un enfoque macro, meso y micro se recurrió a información detallada, producto de investigaciones electrónicas en internet, estudios estadísticos de APESOFT, revistas como Users TI, reportes de gerencia referidos a órdenes de servicios y de compra, cronogramas de proyectos aprobados por los clientes, encuestas, entre otras.

1.1.1. LA INDUSTRIA DE SOFTWARE EN EL MUNDO

En el mundo de las tecnologías de la información no existe una forma de organización global definida para la industria del software que rija su desarrollo o forma de implementación. Cada país, según sus características y posibilidades adopta las vías que considera más favorables. La industria de las tecnologías de la información y las comunicaciones está fuertemente sujeta al desarrollo económico de cada nación y por consiguiente la industria del software como parte de ella se encuentra bajo las mismas condiciones.

Se llama sector o industria del software aquel conformado por unidades económicas cuya actividad principal es la producción, desarrollo y comercialización de programas informáticos.

La actualidad está marcada por una fuerte crisis global que tuvo su comienzo en la esfera inmobiliaria de los Estados Unidos de América y se ha extendido a todo el planeta con consecuencias para todos los países. La industria de las TIC's no es inmune a la crisis y según el Foro Económico Mundial (2009) la industria creció en el 2009 un 2.9 por ciento, por debajo del 4.9 por ciento estimado.

Los países desarrollados siguen llevando el liderazgo en cuanto a desarrollo de las nuevas tecnologías, como puede observarse en la tabla N° 1.1 sobre el ranking de disponibilidad de las nuevas tecnologías lanzado por el Foro Económico Mundial.

Tabla N° 1. 1
Disponibilidad de las nuevas tecnologías 2009-2010

Ranking	Economía	Valor
1	Dinamarca	5.85
2	Suecia	5.84
3	Estados Unidos	5.68
4	Singapur	5.67
5	Suiza	5.58
6	Finlandia	5.53
7	Islandia	5.50
8	Noruega	5.49
9	Holanda	5.48
10	Canadá	5.41

Fuente: Informe global de las tecnologías de la información 2009-2010

Elaboración: Foro Económico Mundial

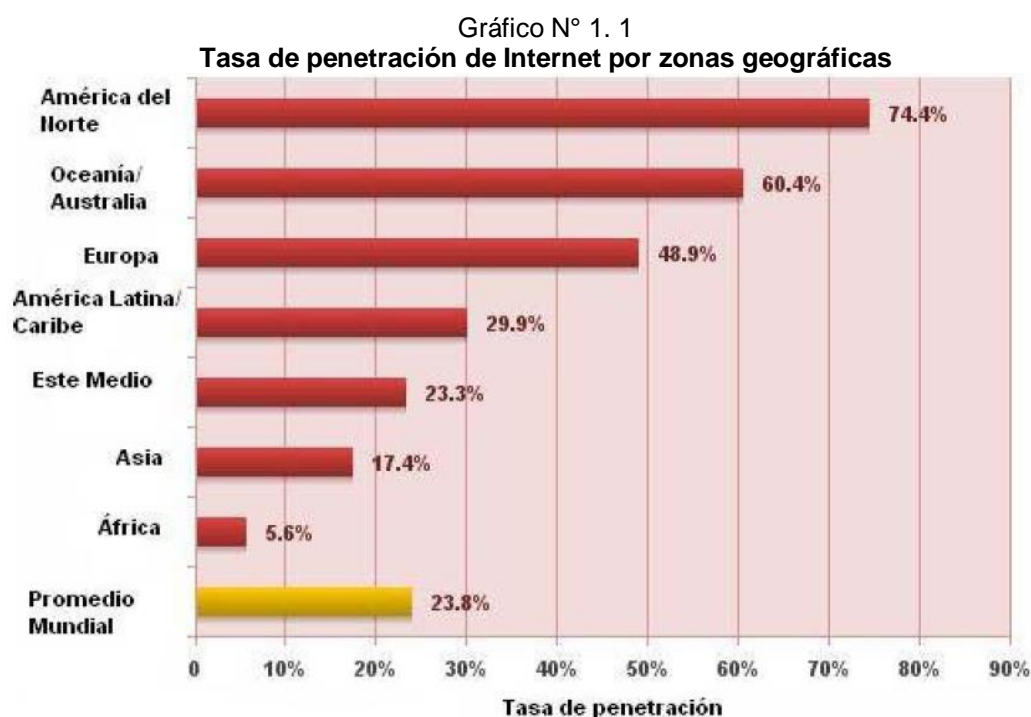
La tabla muestra que para el año comprendido desde 2009 hasta 2010, los diez primeros lugares, de 134 países incluidos, son ocupados por economías del primer mundo y no figura ningún país latinoamericano.

Europa como región sigue ocupando una posición relevante en los rankings en la red de disponibilidad de las nuevas tecnologías y en específico de las infraestructuras para el desarrollo de las nuevas tecnologías. De las 20 primeras posiciones, 12 están ocupadas por países de la región europea, así Suiza en (5to), Finlandia (6to), Islandia (7mo), Noruega (8vo), Holanda (9no), Reino Unido (15to), Austria (16to), Estonia (18vo), Francia (19no) y Alemania (20mo).

En algunos países en desarrollo como Costa Rica, Sri Lanka y Uruguay las exportaciones de software y servicios de TI superan ampliamente los gastos en el interior del país por los mismos conceptos, es decir el gasto interno en software es relativamente bajo. Eso podría indicar que las necesidades de software local están recibiendo menos atención debido a la demanda de los

mercados extranjeros. En otros países como el Brasil, Chile, Kenya y Sudáfrica el gasto interno en software es alto, pero las exportaciones son escasas, lo que parece indicar que hay posibilidades considerables de aumentar las exportaciones. Argentina, Filipinas, India y Malasia se encuentran entre los países de ingresos bajos y medianos que presentan niveles relativamente altos tanto de exportaciones como de ventas de software en el mercado interno. Los logros de China a este respecto son destacables. Según estadísticas oficiales de China, la producción nacional de software pasó de 7.000 millones de dólares en 2000 a 285.000 millones de dólares en 2010. Se calcula que cerca del 90% de esa producción está destinada al mercado interno, aunque por lo general se integra en la fabricación de productos de TIC, y de otros productos, que son luego exportados

En el Gráfico N° 1.1, se observa la tasa de penetración de Internet por zonas geográficas en el mundo.



Fuente: <http://www.internetworldstats.com/stats.htm>

Elaboración: Internet WorldStats. Las tasas de penetración están basadas en una población de 6,710, 029,070 y un estimado de usuarios de Internet de 1,596, 270,108 hasta Marzo, 2009

Del gráfico se observa que América del Norte, Oceanía principalmente Australia y Europa son las tres zonas geográficas que más han incrementado el uso de Internet en sus poblaciones. América del Norte sigue ocupando el primer lugar en lo referido a la tasa de penetración, mientras que África se mantiene en el último lugar con una tasa extremadamente baja.

Todos los indicadores de disponibilidad, uso y desarrollo en las nuevas tecnologías marcan a los países desarrollados a la cabeza, específicamente la región europea y América del Norte. La zona asiática apunta a un ligero incremento en este sentido, pero debido a su alta población los indicadores muestran que aún no son suficientes los resultados obtenidos.

La India ha venido emergiendo como una gran potencia en la producción de software en los últimos tiempos. Ya en 2007, por conceptos de subcontratación de servicios y exportaciones de la industria del software y la información, el monto fue de 17 mil 200 millones de dólares, cifra que se elevó a 60 mil millones anuales para 2010 según Nasscom y McKinsey citados en El Economista de Cuba (2010). De acuerdo con esto, se espera que el crecimiento en las exportaciones indias de TI provenga del mercado de software y de la subcontratación de TI tradicional, como la administración a distancia de sistemas completos, que en la actualidad es un mercado dominado por las grandes empresas de asesoría en TI.

Los países de América Latina no han tenido políticas estatales hechas públicas con vistas a conocer sus pronunciamientos sobre la informatización social, excepto México que presenta un trabajo consecuente desde la década de los 90. Según estudios realizados por Compatia (2009), existe una correlación establecida entre el desarrollo económico y la inversión que realizan los países en Tecnologías de la Información y Software. Los países destinan en promedio 7.5 por ciento de su inversión a la tecnología de Información, mientras en los países latinoamericanos ese promedio es inferior a 2 por ciento.

La industria del software en Latinoamérica tiene una participación del 2.9 por ciento del gasto total en Tecnología de la Información del mundo, siendo Brasil el de mayor participación en ese sentido, como se muestra en la tabla N° 1.2.

Tabla N° 1. 2
América Latina. Participación de los gastos realizados en cada país en el gasto total en el conjunto de países

País	Hardware	Software	Servicios	Gastos internos	Total
Brasil	49%	52%	51%	40%	45%
México	18%	17%	18%	26%	21%
Argentina	10%	11%	10%	7%	10%
Colombia	4%	5%	4%	9%	5%
Venezuela	4%	5%	6%	9%	5%
Chile	3%	3%	4%	5%	4%
Resto	12%	7%	6%	4%	9%

Tomado de: Witsa 2009 (www.witsa.com)

Como se observa en la tabla de la página anterior, Brasil, México y Argentina agrupan tres cuartas partes del gasto de la región.

En América Latina, Brasil resalta como uno de los países que mayores esfuerzos encamina hacia el desarrollo de la industria del software con resultados que lo ubican entre los primeros lugares en este sentido. Cuenta con más de 3 600 empresas desarrolladoras de software.

Cada nación realiza importantes acciones en torno a las nuevas tecnologías debido a que ésta es la industria que marca y determina las pautas para el desarrollo futuro. Las principales economías marchan a la vanguardia en tal sentido, agrandando cada vez más la brecha tecnológica existente y consumiendo las posibilidades de los países menos favorecidos. No solo la gran diferencia económica determina la diferencia tecnológica, políticas internacionales como la explotación de mano de obra barata y el robo de profesionales altamente capacitados es aún un fenómeno cotidiano que atenta contra las naciones de los países tercermundistas o de economías emergentes que puján fuertemente por desarrollar soluciones y alternativas propias. Las grandes empresas que consolidan el poder en las Nuevas Tecnologías se expanden cada vez más, ocupando nichos de mercado que aún quedan disponibles y con sus fuertes inversiones en investigación y desarrollo dejan en posición cada vez menos ventajosa a naciones que tratan de posicionarse en esta industria.

1.1.2. LA INDUSTRIA DE SOFTWARE EN EL PERÚ

Producto del avance tecnológico, los individuos, las empresas, los gobiernos y el comercio viven la era de la globalización. En este marco, una de las industrias que tiene inmensas oportunidades es la del software, cuyo mercado mundial asciende a los \$ 1,500 billones. El Perú es un actor que tiene significativas ventajas para obtener parte del consumo internacional, esto según las últimas estadísticas recogidas por la Asociación Peruana de Software (APESOFT, 2010). La industria de software peruana cuenta con aproximadamente 18 años de existencia, está conformada por 300 empresas formales de las cuales 90% son pequeñas y microempresas, y cuenta con un activo en capital humano de 30 000 programadores de sistemas, genera 6000 puestos de trabajo directo altamente tecnificado y además genera 9000 puestos de trabajo indirecto.

El mercado del software es un sector relativamente joven y de alta especialización, cuyos niveles de inversión son relativamente menores

comparados con otras industrias, pues su principal activo son los recursos humanos con los que cuenta.

En términos generales, una empresa típica de software tiene los siguientes procesos para desarrollar sus aplicaciones: Análisis de requerimientos, Modelamiento de procesos, Modelamiento de datos, Desarrollo y Prueba y entrega.

Sin embargo, existen en menor porcentaje empresas que adaptan una metodología de desarrollo de software de acuerdo a sus propias características y necesidades.

La tabla N° 1.3 refleja que las empresas inmersas en el desarrollo de software no están especializadas en una sola actividad sino que se complementan con otras actividades de servicios para poder operar.

Tabla N° 1. 3
¿A qué se dedica la empresa?

Actividades	%
Desarrollo a medida	15.82
Consultoría de sistemas	15.30
Fabricante de software	15.16
Integrador de sistemas	12.80
Comercializador y distribuidor de software	12.03
Servicios informáticos diversos	11.90
Outsourcing	10.19
Servicios de internet	6.79

Fuente: Encuesta realizada por Prompex Perú y Apesoft a 150 empresas del sector (2011)

Elaboración: Prompex Perú y Apesoft

De la tabla se destaca el desarrollo de soluciones a medida, la consultoría de sistemas, la fabricación de software, entre otras.

La tabla N° 1.4 se refiere al uso de plataformas tecnológicas y lenguajes de programación.

De la tabla de la página siguiente, en materia de lenguajes de programación más utilizados se tiene a la plataforma Microsoft, como la más preferida por los desarrolladores. Seguida de cerca por Java cuyo lenguaje va captando más adeptos.

Tabla N° 1. 4
Uso de plataformas tecnológicas y lenguajes de programación

MARCA	FABRICANTE	%
Visual Studio .Net	Microsoft	35.41
Java	Sun Microsystems	20.32
Oracle Developer	Oracle	12.80
C++ / Visual Basic 6.0	Microsoft	9.82
Power Builder	Sybase Inc	5.91
Visual Fox Pro	Microsoft	3.96
PHP	Código libre	2.64
Otros		9.14

Fuente: Encuesta realizada por Prompex Perú y Apesoft a 150 empresas del sector (2011)

Elaboración: Prompex Perú y Apesoft

La tabla N° 1.5 se basa en los sistemas gestores de base de datos más utilizados en el mercado.

Tabla N° 1. 5
Gestores de Base de Datos más utilizados

MARCA	FABRICANTE	%
SQL Server	Microsoft	37.79
Oracle	Oracle	22.91
Access	Microsoft	14.52
DB2	IBM	6.08
Sybase	Sybase Inc	5.73
Informix	IBM	5.08
Visual Fox Pro	Microsoft	3.02
Otros		4.87

Fuente: Encuesta realizada por Prompex Perú y Apesoft a 150 empresas del sector (2011)

Elaboración: Prompex Perú y Apesoft

De la tabla anterior, en materia de base de datos los dos más utilizados son el SQL Server de Microsoft con 37.79%, seguido por Oracle con 22.91%, entre otros.

En la tabla N° 1.6 se visualiza el salario promedio mensual en dólares para los involucrados en el desarrollo de un proyecto de software.

De la tabla de la página siguiente, se aprecia que en general los jefes de proyectos tienen un sueldo promedio de U\$ 1,570, los analistas tienen una remuneración promedio de U\$ 1,144 y los programadores de U\$ 800, cabe resaltar que los sueldos han tenido incrementos en comparación con el año

2008, esto se puede deber al incremento de las ventas del mercado. Si la tendencia sigue en alza los sueldos tenderían a seguir incrementándose para el 2013.

Tabla N° 1. 6
Salario promedio mensual en dólares - 2011

Salario Promedio Mensual en dólares - 2011				
Tamaño de Empresa	Jefe de Proyectos	Analista	Programador	Total
Grande	\$2,214.29	\$1,628.57	\$1,192.86	\$5,035.72
Mediana	\$1,582.50	\$1,197.50	\$799.00	\$3,579.00
Pequeña	\$1,348.04	\$957.75	\$653.75	\$2,959.54
Micro Empresa	\$1,138.54	\$792.87	\$555.31	\$2486.72

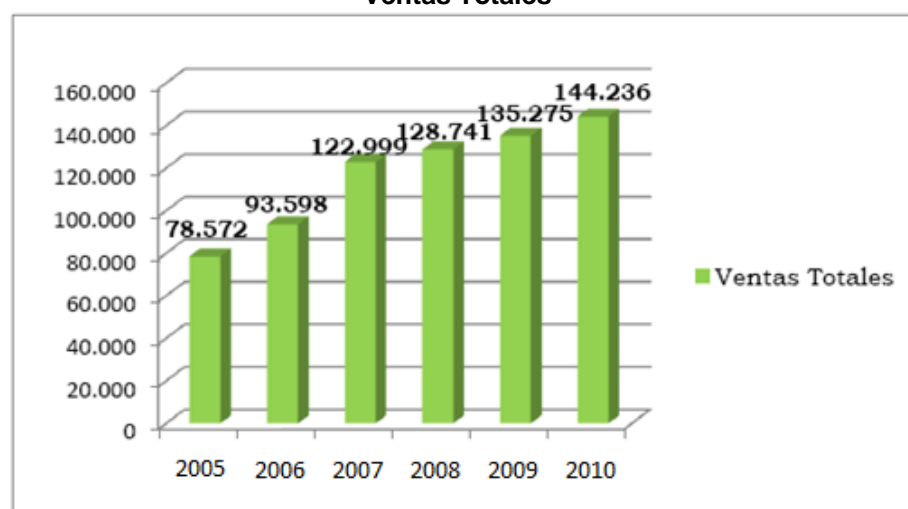
Fuente: Encuesta realizada por Prompex Perú y Apesoft a 150 empresas del sector (2011)

Elaboración: Prompex Perú y Apesoft

La industria del software es eminentemente nacional, pues por ejemplo en el año 2010, únicamente el 8% de las empresas eran financiadas con capitales internacionales. Además, el 76% de las empresas nacionales tienen menos de 18 años de operaciones.

El gráfico N° 1.2 muestra los ingresos por ventas totales durante los años 2005 al 2010, los cuales reflejan una tendencia al crecimiento para el sector de software peruano.

Gráfico N° 1. 2
Ventas Totales



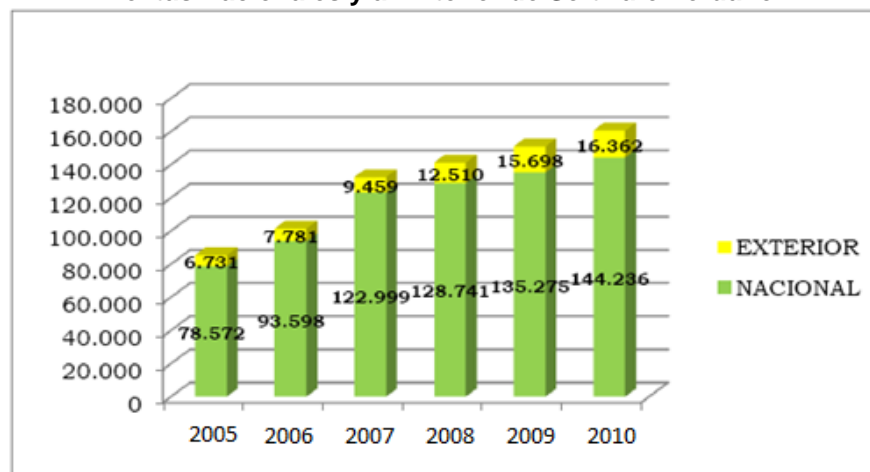
Fuente: Cuore CCR (2010)

Elaboración: CCR

El gráfico refleja que para el año 2010, el estimado de ingresos del sector fue de 144,2 millones de dólares.

El gráfico N° 1.3 muestra las ventas hacia el mercado nacional y hacia el exterior de software peruano.

Gráfico N° 1. 3
Ventas Nacionales y al Exterior de Software Peruano



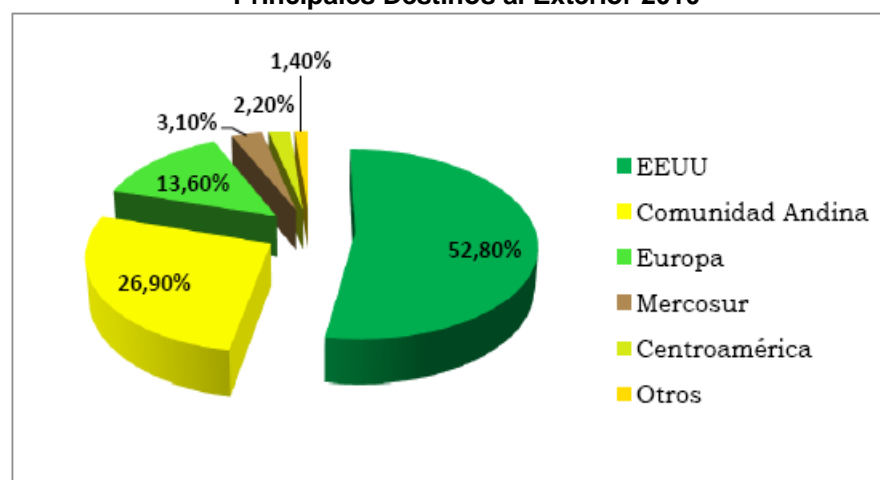
Fuente: Cuore CCR (2010)

Elaboración: CCR

El gráfico refleja que la industria produce básicamente para el mercado local, destinando el 90% de sus ventas, en tanto que el 10% restante se destina al mercado externo. Sin embargo, las exportaciones en el 2009 tuvieron un crecimiento de más del 25% respecto a las exportaciones del año 2008.

El gráfico N° 1.4 hace alusión a un estudio realizado por Cuore CCR (2010), por encargo de PACIS (Programa de Apoyo a la Competitividad de la Industria del Software) y APESOFT, sobre los principales destinos de exportación de software peruano para el año 2010.

Gráfico N° 1. 4
Principales Destinos al Exterior 2010



Fuente: Cuore CCR (2010)

Elaboración: CCR

Dicho estudio menciona que el 25% de las empresas exporta sus productos al exterior, siendo los principales mercados: EEUU, la Comunidad Andina y Europa, lo cual se observa en el gráfico de la página anterior.

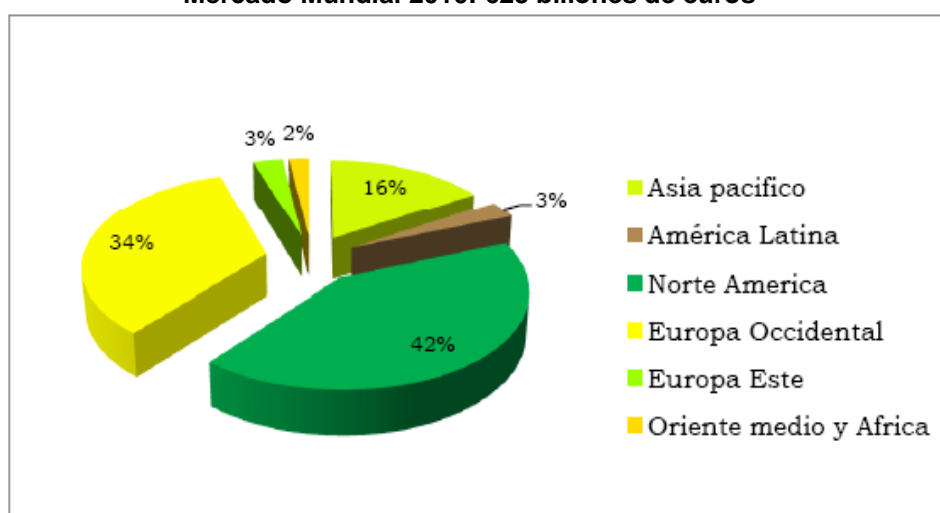
Aun cuando el sector de software todavía representa una pequeña fracción de la producción nacional (0,6% del PBI), su crecimiento es de suma importancia, pues permite una mejora constante en la productividad de los demás sectores productivos. La industria de software promueve una mayor competitividad en costos, tiempo, y una mayor accesibilidad a la información, resultando ser una pieza fundamental en el desarrollo de la sociedad de información.

El software producido en el país es principalmente software aplicativo de carácter horizontal, por ejemplo dirigido a los sistemas de contabilidad, personal, caja, logística, etc. Sin embargo, existen empresas que desarrollan software a la medida de las necesidades del cliente y servicios tales como el outsourcing y consultoría en sistemas.

Otra clase de software producido en el Perú, son los orientados a diseñar y desarrollar software a sectores especializados, tales como: el sector público, bancario, salud, transporte marítimo, etc.

El gráfico N° 1.5 proporciona una figura estadística que muestra que el mercado mundial de software, se encuentra concentrado en América del Norte, Europa Occidental, y Asia Pacífico. Lo que debería incentivar a las empresas peruanas a exportar a esos mercados.

Gráfico N° 1. 5
Mercado Mundial 2010: 625 billones de euros



Fuente: Cuore CCR (2010)
Elaboración: CCR

Del gráfico de la página anterior, se deduce que la cercanía y los tratados comerciales del Perú con los principales mercados de software podrían ser considerados como una ventaja competitiva. Estados Unidos y Japón representan el 38% y 10% del mercado mundial respectivamente.

El Perú puede ser considerado como un destacado nuevo competidor por su continua expansión y crecimiento acelerado. Pues aun cuando algunos países disminuyeron sus ingresos en esta industria debido a la crisis económica del 2008, el Perú mantuvo su crecimiento a un ritmo de 7%, comparado al año 2007. Es en este contexto que el gobierno peruano viene lanzando iniciativas que fomentan la industria, como el plan CODESI, plan de desarrollo de las tecnologías de información de CONCYTEC, asimismo es fundamental mencionar el programa PACIS, respaldado por el Banco Interamericano de Desarrollo (BID), cuyo objetivo es promover la acreditación de CMMI de las empresas nacionales prestadoras de servicio de software. El programa PACIS cuenta con 90 empresas beneficiarias, siendo que el 10% ya cuenta con la certificación, lo cual ya puede ser considerado un logro, pues hasta el año 2006, en el Perú no existía ninguna empresa con dicha certificación.

Se destaca a su vez, la iniciativa de la Comisión de Promoción del Perú para la Exportación y el Turismo – PromPerú mediante el nacimiento de CREA, Software Perú, el cual reúne un grupo de empresas peruanas de software y proveedoras de servicios de tecnologías de la información (TI) con el objetivo de impulsar la competitividad internacional de la industria peruana, en la actualidad está conformado por un grupo de 29 empresas.

1.1.3. UNIDAD DE TECNOLOGÍAS DE INFORMACIÓN DE LA EMPRESA CCJ

Corporación Comercial Jerusalem S.A.C., denominada en adelante CCJ S.A.C., es una organización que inició sus operaciones en noviembre de 2004 y fue creada con la visión de servicio en el diseño, desarrollo y ejecución de Proyectos de ingeniería, consultoría, supervisión y asesoría técnica.

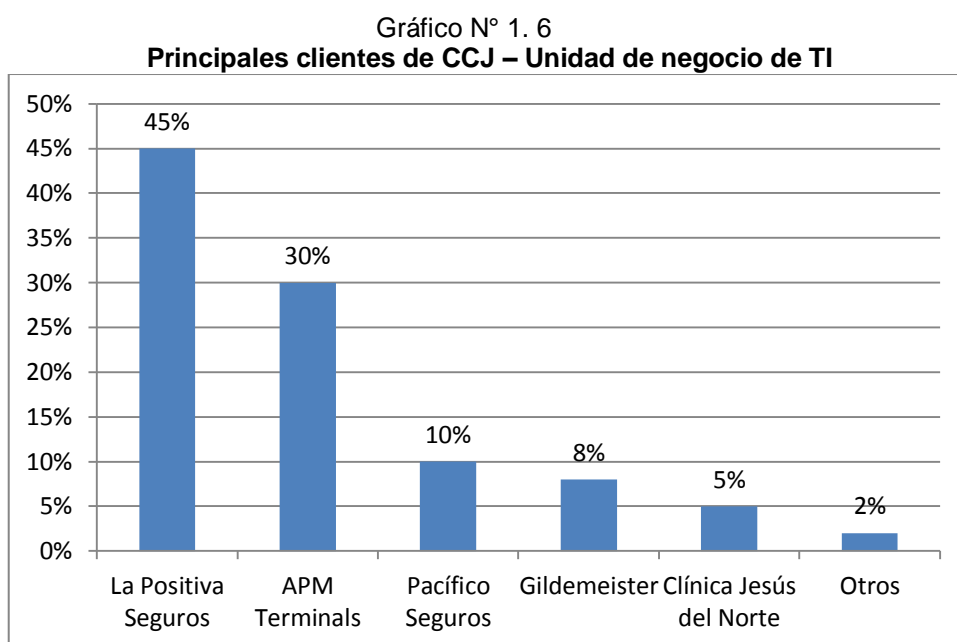
La corporación CCJ está compuesta por cuatro unidades de negocio: Infraestructura, Mantenimiento, Telecomunicaciones y Tecnologías de Información.

La unidad de negocio de Tecnologías de Información, en adelante TI, es donde se ponen en marcha todos los proyectos de desarrollo software y mantenimiento de servidores que llegan a la empresa.

1.1.3.1. PRINCIPALES CLIENTES DE LA UNIDAD DE NEGOCIO DE TI

La unidad de negocio de TI funciona como proveedora de servicios informáticos mediante el desarrollo de software a medida y consultoría de sistemas. Tiene como principales clientes a La Positiva Seguros S.A., Pacífico Seguros S.A., APM Terminals S.A., Automotores Gildemeister Perú S.A., entre otros.

El gráfico N° 1.6 muestra los principales clientes de CCJ para la unidad de negocio de TI.



Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

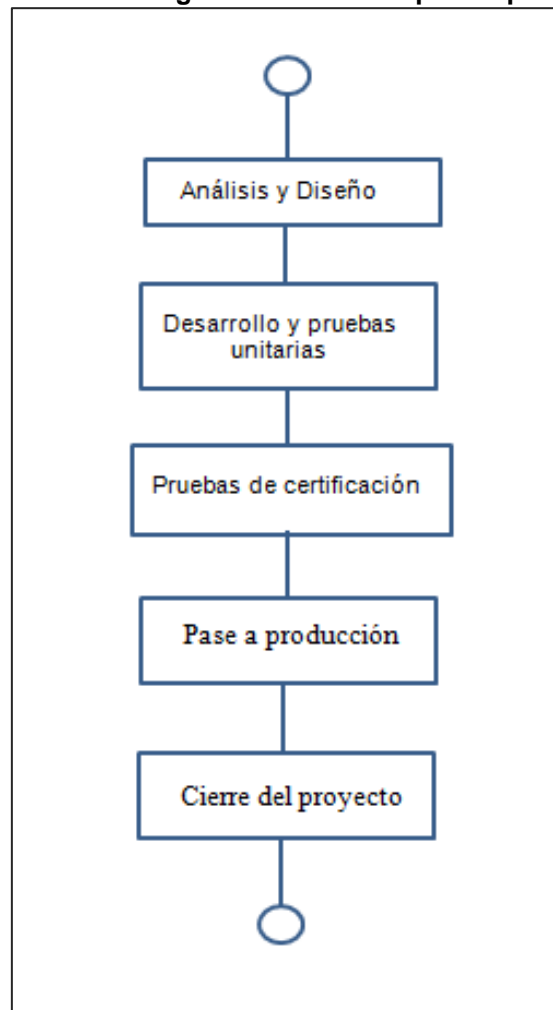
Del gráfico, el cliente que solicita con mayor frecuencia los servicios de la unidad de negocio de TI es la Positiva Seguros con un 45%, seguida de APM Terminals Callao con un 30%, después Pacífico Seguros con un 10%, entre otras.

CCJ pretende alcanzar niveles altos de calidad mediante las certificaciones en tecnologías Microsoft brindadas a su personal de TI, por ser las tecnologías más requeridas en los proyectos desarrollados. Entre las diversas herramientas de desarrollo que maneja, se encuentran: Visual Basic 6.0, Visual Studio .Net 2010, SQL Server 2008 R2, Toad for Oracle 11g, BizTalk Server 2009, etc.

1.1.3.2. METODOLOGÍA DE DESARROLLO EMPLEADA EN LA UNIDAD DE NEGOCIO DE TI

En el gráfico N° 1.7 se visualiza la metodología de desarrollo de software empleada actualmente, a través de un flujo, en la unidad de negocio de Tecnologías de Información.

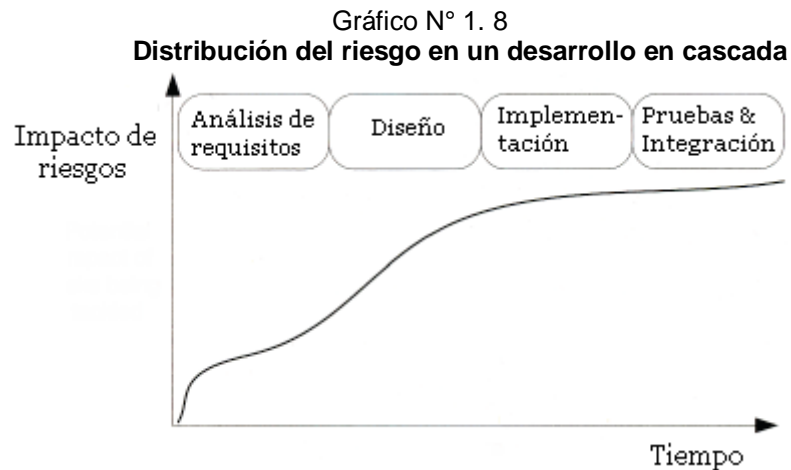
Gráfico N° 1. 7
Flujo de la Metodología de software empleada por CCJ – TI



Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico, la metodología de desarrollo de software comprende las fases de Análisis y diseño, desarrollo y pruebas unitarias, pruebas de certificación, pase a producción, y cierre del proyecto. Este es un modelo adaptado del denominado Ciclo de Vida en Cascada.

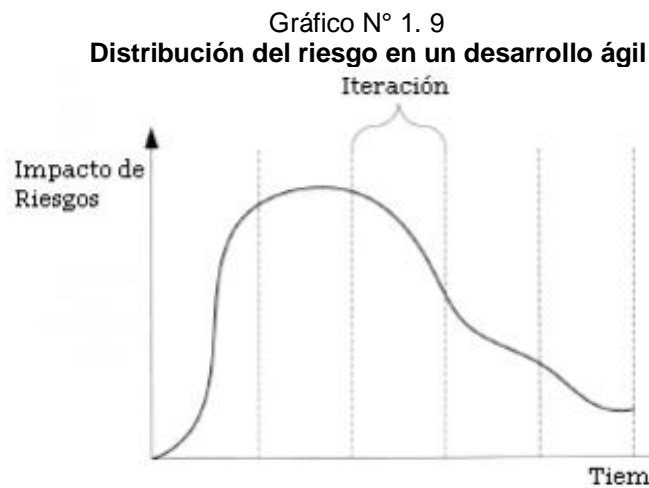
El gráfico N° 1.8 muestra la distribución del riesgo en un desarrollo en cascada.



Basado en: Rodríguez González, Pilar (2008). Estudio de la Aplicación de Metodologías Ágiles para la Evolución de productos software. Madrid, España. Pág. 11.

Del gráfico, una de las dificultades del uso de metodologías convencionales como la de cascada es la lentitud del proceso de desarrollo. Asimismo, las metodologías convencionales tienden a acumular los riesgos y dificultades que surgen en el desarrollo del producto al final del proyecto, repercutiendo en retrasos en la entrega de productos o influyendo en la incorrecta ejecución de las últimas fases del ciclo de vida.

El gráfico N° 1.9 muestra la distribución del riesgo en un desarrollo ágil.



Basado en: Rodríguez González, Pilar (2008). Estudio de la Aplicación de Metodologías Ágiles para la Evolución de productos software. Madrid, España. Pág. 12.

Del gráfico, se puede deducir que una metodología ágil se basa en el desarrollo en ciclos de corta duración lo cual favorece a que los riesgos y dificultades se repartan a lo largo del desarrollo del producto,

principalmente al comienzo del mismo y permite ir aprendiendo de estos riesgos y dificultades.

El gráfico N° 1.10 refleja la cultura Scrum frente a la cultura del convencional ciclo de vida en Cascada.

Gráfico N° 1. 10
Dos culturas, Desarrollo Convencional vs. Scrum



Fuente: Tesis de Máster en TI - Universidad Politécnica de Madrid

Elaboración: Rodríguez González Pilar

Del gráfico se deduce que, la efectividad de la metodología para la gestión de proyectos se basa en un conjunto de valores fundamentales que deben seguir todos los integrantes del equipo, principios sobre los que reposan el resto de prácticas: compromiso, esmero, franqueza, respeto y valor.

1.1.3.3. INDICADORES DE PRODUCTIVIDAD EN EL PROCESO DESARROLLO DE SOFTWARE

Actualmente, la productividad del desarrollo de software en la unidad de negocio de TI se ha visto afectada debido al incumplimiento de los tiempos propuestos para la culminación de los proyectos. Lo cual conlleva a un incremento en los costos iniciales estimados por proyecto, que generalmente son asumidos por la Empresa CCJ, generando así pérdidas en los ingresos de la unidad de TI.

Las métricas en el desarrollo de software, son medidas efectuadas sobre los programas, documentación, su desarrollo y mantenimiento, o sobre algún aspecto del sistema en desarrollo del proceso empleado que permite, previa comparación con unos valores (medidas) de

referencia, obtener conclusiones sobre el aspecto medido con el fin de adoptar las decisiones necesarias.

El proceso de planificación del desarrollo de cualquier sistema debe hacerse partiendo de una estimación del trabajo a realizar, Sólo a partir de ello es factible conocer los recursos necesarios y el tiempo necesario para su realización. La estimación precisa de ciertas métricas como el esfuerzo de desarrollo es indispensable para la adecuada planificación de las actividades de desarrollo y mantenimiento.

Las métricas se utilizan para evaluar y controlar el proceso de desarrollo de software, de forma que permitan:

- Indicar la calidad del producto
- Evaluar la productividad de los desarrolladores
- Evaluar los beneficios (en cuanto a calidad y productividad)
- Derivados del uso de nuevos métodos y herramientas de ingeniería del software
- Establecer una línea base para la estimación
- Justificar el uso de nuevas herramientas o de formación adicional

Los tipos de métricas que existen son; en primer lugar, Métricas del producto (entrada código fuente), que comprende: Tamaño, estructura de datos y lógica de procedimiento, y en segundo lugar, Métricas del proceso (entorno de desarrollo), que comprende: Tiempo de desarrollo, reusabilidad y productividad.

Las métricas incluyen como objeto de medición:

1. La cantidad de esfuerzo necesaria para desarrollar un sistema
2. La duración del proyecto
3. El tamaño y la volatilidad de los requerimientos
4. El costo global del proyecto, el tipo de métrica que se recomienda incluye a las siguientes: Tamaño del software, puntos de función y cuentas de objetos y métodos)
5. El esfuerzo del trabajo del personal: Esfuerzo real medido en unidades persona/mes y Esfuerzo reportado en unidades persona/mes.
6. Volatilidad de los requerimientos (cambios de los requerimientos)

7. Experiencia del dominio o aplicación (de la arquitectura de desarrollo utilizada, de las herramientas y métodos empleados, años globales de experiencia en el desarrollo)
8. Rotación de personal

La tabla N° 1.7 permite visualizar la relación entre los factores de calidad del software y las métricas generalmente empleadas en los proyectos de desarrollo de software.

Tabla N° 1. 7
Relación entre los factores de calidad del software y las métricas

<div>Métrica de la calidad del software</div> <div>Factor de calidad</div>	Corrección	Fiabilidad	Eficiencia	Integridad	Mantenimiento	Flexibilidad	Capacidad de pruebas	Portabilidad	Reusabilidad (capacidad de reutilización)	Interoperabilidad	Usabilidad (facilidad de manejo)
Facilidad de auditoria				*			*				
Exactitud		*									
Estandarización de comunicaciones										*	
Compleción		*			*	*					
Complejidad	*										
Concisión	*	*				*	*				
Consistencia				*	*	*					
Estandarización de datos	*	*				*	*			*	
Tolerancia a errores		*									
Eficiencia de ejecución			*								
Capacidad de expansión						*					
Generalidad						*			*	*	*
Independencia del hardware								*	*		
Instrumentación					*	*	*				
Modularidad	*					*	*	*	*	*	
Operatividad			*								*
Seguridad			*								
Autodocumentación					*						
Simplicidad					*	*	*		*	*	
Independencia del sistema		*					*	*	*		
Trazabilidad								*	*		
Facilidad de formación											

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla, el peso que se asigna a cada métrica depende de los productos y negocios locales.

La tabla N° 1.8 muestra los datos obtenidos de la Gerencia de Tecnologías de Información acerca de los principales indicadores de productividad de cada desarrollador medidos en el periodo de mayo a diciembre de 2013.

Tabla N° 1. 8
Indicadores de productividad por desarrollador empleado en CCJ S.A.C.
Mayo – Diciembre 2013

MÉTRICAS DE LA CALIDAD DEL SOFTWARE	DEL PRODUCTO			DEL PROCESO		
	Tamaño (líneas código)	Estructura Datos (cantidad de datos)	Lógica de procedimiento (nivel de conocimiento)	Tiempo de desarrollo (horas - hombre)	Reutilización (código nuevo vs. reusado)	Productividad (salidas / entradas)
Desarrollador 1	0.8	0.8	0.5	0.8	0.7	0.72
Desarrollador 2	0.6	0.7	0.4	0.9	0.6	0.64
Desarrollador 3	0.5	0.6	0.3	0.6	0.4	0.48
Desarrollador 4	0.3	0.5	0.2	0.5	0.2	0.34

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Gerencia de Tecnología de Información

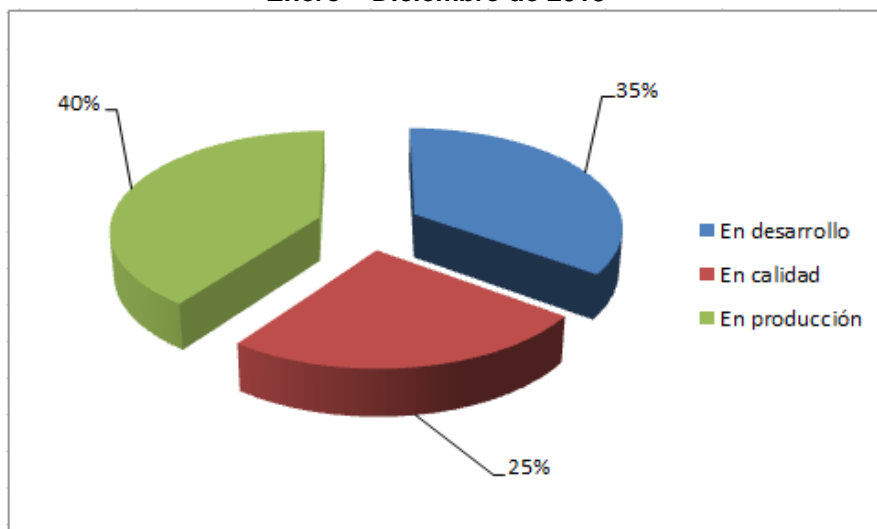
La tabla refleja que los indicadores de productividad, de la unidad de negocio de TI, no son los más óptimos. Ello debido principalmente a:

- ✓ El insuficiente dominio de la metodología de desarrollo de software empleado en CCJ.
- ✓ La carencia de esfuerzo y compromiso de trabajo por parte del personal.
- ✓ Una estimación inicial incorrecta lo que genera volatilidad de requerimientos.
- ✓ La insuficiente experiencia o nivel de conocimiento de la tecnología o lenguaje de programación.
- ✓ La rotación de personal y su inadecuada inducción a los proyectos nuevos o pendientes.

En el gráfico N° 1.11 se observa el estado de los proyectos de software que se desarrollaron en la Unidad de TI para el año 2013.

Del gráfico de la página siguiente, el sector circular de color azul muestra que del total de proyectos un 35% se encuentra en estado de Desarrollo, el sector circular rojo refleja que un 25% está en el ambiente de Calidad (QA), y el de color verde muestra que sólo un 40% están en Producción. Es decir, aún se están arrastrando proyectos desde el año anterior.

Gráfico N° 1. 11
Estado de los Proyectos de Software desarrollados en CCJ S.A.C.
Enero – Diciembre de 2013



Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

En la tabla N° 1.9 se observa una lista de los principales proyectos de software que fueron ejecutados en el año 2013 en CCJ.

Tabla N° 1. 9
Lista de Proyectos de Software que debían ser ejecutados en CCJ S.A.C.
Enero - Diciembre de 2013

Ítem	Nombre del proyecto	Fecha de inicio	Fecha de término	Duración	Estado
001	Cargar Base de Datos del Sistema de Procuración de la Suite Jubilare	01/02/2013	14/03/2013	30 días	En producción
002	APM Terminals Implementación ESB	26/06/2013	09/09/2013	54 días	En producción
003	Generación de un número único de documento que agrupe las proformas LPG, LPV y LPC	27/06/2013	09/08/2013	32 días	En calidad
004	Mantenimiento de Servidores BizTalk	04/11/2013	29/11/2013	20 días	En desarrollo
005	Creación de ambiente de calidad BizTalk Servidor de aplicaciones y base de datos	02/12/2013	13/12/2013	10 días	En desarrollo

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico, se contempla cinco proyectos cada uno con una fecha de inicio y fecha de término definidas, así como su duración respectiva en días de trabajo (se considera un día de trabajo equivalente a 8 horas de trabajo diario de lunes a viernes).

En la tabla N° 1.10 se observa la duración, los retrasos y el estado de cada proyecto mostrado en la lista proporcionada en la tabla N° 1.9.

Tabla N° 1. 10
Duración, Retraso y Estado de los Proyectos de Software en CCJ S.A.C.
Enero - Diciembre de 2013

Ítem	Nombre del proyecto	Fecha de inicio	Fecha de término	Duración	Retraso	Estado
001	Cargar Base de Datos del Sistema de Procuración de la Suite Jubilare	01/02/2013	14/03/2013	30 días	20 días	En producción
002	APM Terminals Implementación ESB	26/06/2013	09/09/2013	54 días	30 días	En producción
003	Generación de un número único de documento que agrupe las proformas LPG, LPV y LPC	27/06/2013	09/08/2013	32 días	50 días	En calidad
004	Mantenimiento de Servidores BizTalk	04/11/2013	29/11/2013	20 días	5 días	En desarrollo
005	Creación de ambiente de calidad BizTalk Servidor de aplicaciones y base de datos	02/12/2013	13/12/2013	10 días	3 días	En desarrollo

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla, se puede notar que existe un desfase en días en los tiempos propuestos para el término de un proyecto. Fíjese en el recuadro sombreado de anaranjado que indica el retraso en días y en la columna siguiente el estado del mismo respectivamente, el ítem 003 cuyo desfase es de 50 días (equivalente a 400 horas de trabajo perdido considerando que se trabaja 8 horas diarias de lunes a viernes) y aún se encuentra en QA, lo cual que la metodología de desarrollo empleada no está siendo efectiva del todo.

La tabla N° 1.11 muestra los tiempos de asignación versus los tiempos realmente utilizados por cada desarrollador en el proyecto especificado.

Tabla N° 1. 11
Cuadro de Tiempo Asignado vs. Tiempo Utilizado por Desarrollador

Ítem	Nombre del Proyecto	Número de Desarrollador	Tiempo Asignado (días)	Tiempo Utilizado (días)	Tiempo perdido (días)	Tiempo perdido (horas)
001	Cargar Base de Datos del Sistema de Procuración de la Suite Jubilare	Desarrollador 1	15	25	10	80
		Desarrollador 2	15	25	10	80
002	APM Terminals Implementación ESB	Desarrollador 1	30	45	15	120
		Desarrollador 2	24	39	15	120
003	Generación de un número único de documento que agrupe las proformas LPG, LPV y LPC	Desarrollador 1	13	46	30	240
		Desarrollador 2	13	36	20	160
		Desarrollador 3	6	6	0	0
004	Mantenimiento de Servidores BizTalk	Desarrollador 1	20	25	5	40
005	Creación de ambiente de calidad BizTalk Servidor de aplicaciones y base de datos	Desarrollador 1	7	10	3	24
		Desarrollador 2	3	3	0	0

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

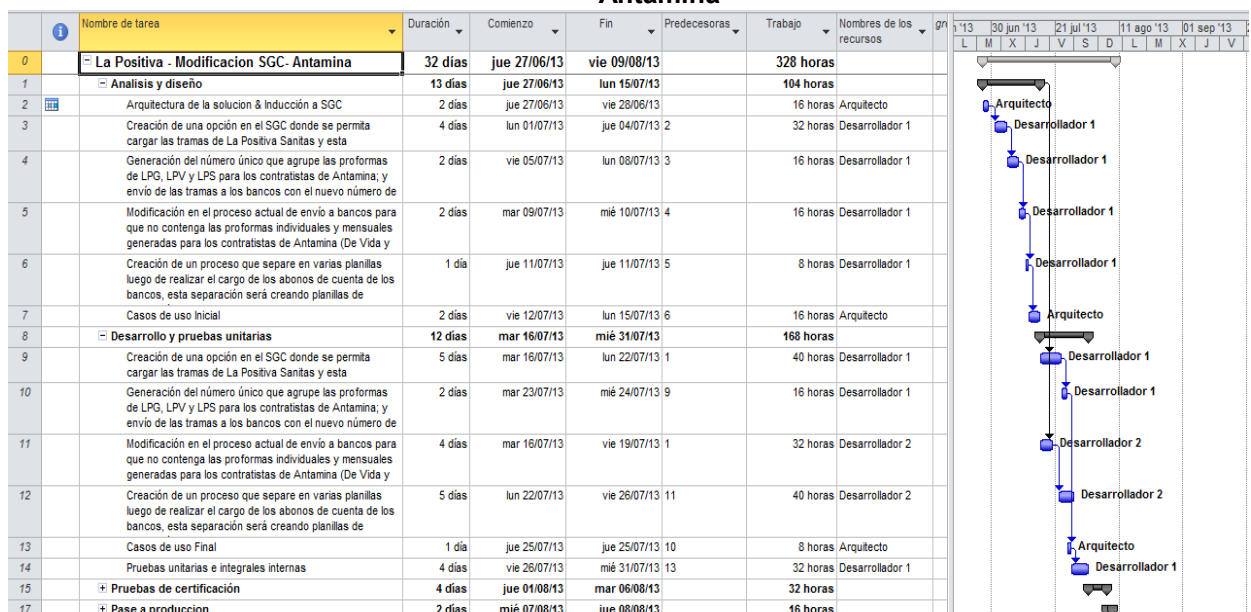
Elaboración: Propia

De la tabla de la página anterior, se observa que existe gran brecha entre el tiempo asignado en contraposición con el tiempo utilizado por cada desarrollador para la culminación de los módulos del proyecto. Fíjese en la columna Tiempo perdido en horas, la cual muestra los tiempos excedidos por cada desarrollador en horas, ello genera un costo añadido al proyecto que no se contemplaron inicialmente en la propuesta técnica.

Como se puede analizar ningún proyecto llevado a cabo durante el año 2013 se ha podido culminar en el tiempo establecido en La Propuesta Técnica propuesto por CCJ, lo cual se refleja en la insatisfacción de los clientes.

El gráfico N° 1.12 permite visualizar el cronograma del proyecto “Generación de un número único de documento que agrupe las proformas LPG, LPV y LPC”, el cual inició el 27/06/2013 y debía haber terminado el 09/08/2013.

Gráfico N° 1. 12
Cronograma del proyecto “La Positiva – Modificación Sistema de Gestión de Cobranzas - Antamina”



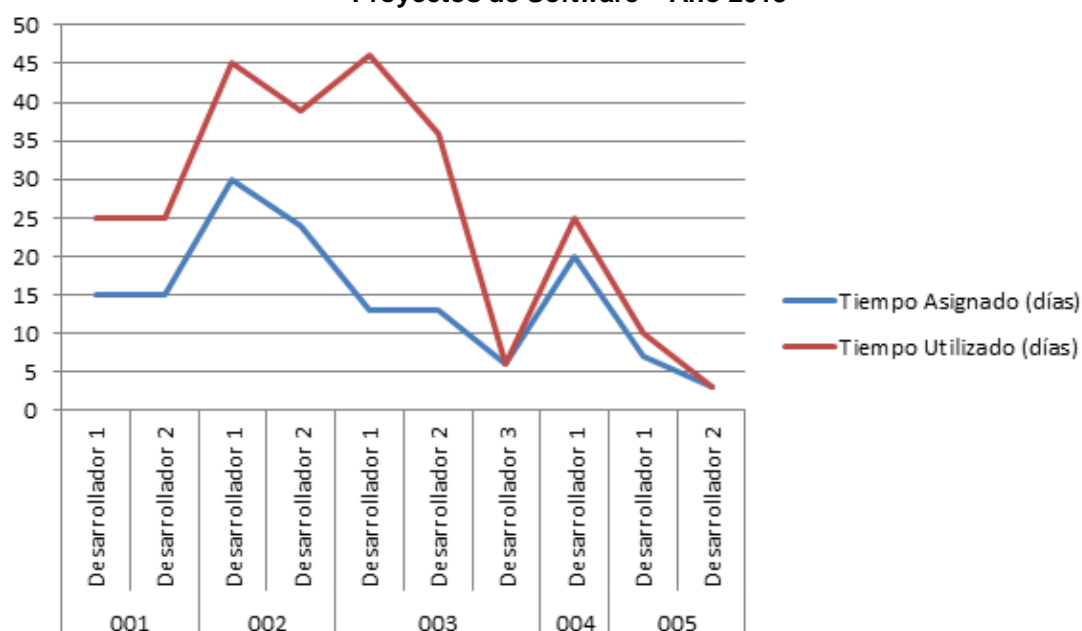
Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

El cronograma anterior no se llegó a cumplir, debido a que en la fase de Análisis y Diseño, primera fase contemplada en la metodología de desarrollo actual, no se definió correctamente el Análisis Funcional versión 1.0, el cuál fue aprobado y aceptado por todas las partes, y que inicialmente consideraba cuatro requerimientos del sistema. Pero

a medida que se desarrollaba el proyecto se fueron cambiando los requerimientos del sistema hasta llegar a tener un Análisis Funcional versión 7.0. Ello es un causante del retraso del proyecto, el cuál hasta la fecha de hoy no ha pasado a producción encontrándose en el ambiente de calidad. Lo cual evidencia que los tiempos asignados no están siendo bien estimados afectando la fase de Desarrollo y la fase de Pruebas consecuentemente.

El gráfico N° 1.13 evidencia los desfases de tiempo en días por desarrollador y por proyecto.

Gráfico N° 1. 13
Gráfico lineal de Tiempo Asignado vs. Tiempo Utilizado en los Proyectos de Software – Año 2013



Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

El gráfico de los proyectos de software realizados durante el año 2013, muestra que la curva lineal del tiempo real utilizado es muy superior a la curva del tiempo asignado a cada desarrollador. Lo cual refleja una vez más que se está estimando de manera incorrecta los tiempos de desarrollo asignados a cada desarrollador en los proyectos.

La tabla N° 1.12 permite percibir la insatisfacción de los trabajadores involucrados en el desarrollo de los proyectos de software en CCJ (Ver la encuesta aplicada en el Anexo II).

De la tabla de la página siguiente, se percibe un desacuerdo generalizado con la metodología empleada actualmente, la cual no permite optimizar tiempos ni costos, y que tampoco permite un manejo

adecuado de las líneas de comunicación ni genera un buen clima laboral. Fíjese que a la pregunta ¿Estaría de acuerdo con la propuesta de implementar una metodología ágil para el desarrollo de software? El 70% de los trabajadores está muy de acuerdo, mientras que un 25% está de acuerdo, y sólo un 5% poco de acuerdo.

Tabla N° 1. 12
Cuadro que refleja la insatisfacción de los trabajadores

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	En desacuerdo
¿Está de acuerdo con la metodología de desarrollo de software usada actualmente?	-	10%	30%	60%
¿Está de acuerdo en que ésta metodología permite optimizar tiempos y costos?	-	5%	15%	80%
¿Está de acuerdo en que ésta metodología permite mejorar las líneas de comunicación?	-	3%	17%	80%
¿Está de acuerdo en que ésta metodología permite la existencia de un adecuado clima laboral?	-	1%	19%	80%
¿Estaría de acuerdo con la propuesta de implementar una metodología ágil para el desarrollo de software?	70%	25%	5%	-

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

La tabla N° 1.13 permite percibir la insatisfacción de los clientes quiénes hacen uso de los servicios de tecnologías de información de la Unidad de TI (Ver la encuesta aplicada en el Anexo II).

Tabla N° 1. 13
Cuadro que refleja la insatisfacción de los clientes

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	Muy en desacuerdo
¿Está de acuerdo con los plazos establecidos para su proyecto?	5%	55%	40%	-
¿Está de acuerdo con los costos establecidos para su proyecto?	2%	60%	30%	8%
¿Cree Ud. que la empresa cuenta con tecnologías de vanguardia?	-	55%	35%	10%
¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto?	-	5%	25%	70%
¿Está de acuerdo en que la empresa termina su proyecto en los plazos y tiempos establecidos?	8%	22%	50%	20%
¿Está de acuerdo que cuando la empresa culmina su proyecto Ud. sale conforme?	10%	25%	50%	15%

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla se percibe que existe un desacuerdo por parte de los clientes o usuarios finales con los plazos establecidos para la realización de un proyecto, además un 70% está muy en desacuerdo ante la pregunta ¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto? Fíjese que en la fila cinco de la tabla, un 50% está poco de acuerdo acerca de que la empresa termina sus proyectos en los plazos y tiempos establecidos y que también un 50% está poco de acuerdo acerca de que cuando culminan los proyectos el cliente se va conforme.

La tabla N° 1.14 muestra los datos que la encuesta arrojó acerca de las razones por las cuales la empresa no se decide a cambiar la metodología de desarrollo de software actual (Ver la encuesta aplicada en el Anexo II).

Tabla N° 1. 14
Razones por las que no cambiarían la metodología de desarrollo

Ítem	%
Por desconocimiento de la nueva metodología	35%
Por las limitaciones financieras	25%
La falta de asesoramiento en la nueva metodología	19%
No se está preparado para la adopción de la nueva metodología	13%
Es uno de los objetivos y próximamente se logrará	5%
No está entre los planes inmediatos	2%
Otros	1%

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla se visualiza que la razón principal para que no se cambie de metodología de desarrollo es por el desconocimiento de la nueva metodología y por consiguiente el miedo a los resultados que se obtendrían si se pondría en práctica una metodología ágil, en este caso Scrum.

El mercado actual es altamente competitivo y la tecnología es muy cambiante. En el desarrollo de un proyecto de software se pide básicamente rapidez, calidad y reducción de costos, y más aún en CCJ que necesita que sus desarrolladores tengan la habilidad de migrar de una tecnología a otra en el menor tiempo posible, pero para asumir estos retos es necesario tener agilidad y flexibilidad.

Todo ello resume las principales debilidades y falencias que existen en la unidad de negocio de TI de la empresa CCJ, traducidas en un problema de inadecuada asignación de tiempos lo que conlleva a las demoras para la finalización de un proyecto y consecuentemente a un incremento de los costos de producción, generando así baja productividad.

1.2. FORMULACIÓN DEL PROBLEMA

PROBLEMA GENERAL

- ¿Cómo influye la Metodología Scrum sobre el incremento de la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C. Lima?

1.3. OBJETIVOS

OBJETIVO GENERAL

- Determinar la influencia de la Metodología Scrum sobre el incremento de la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C. Lima.

1.4. JUSTIFICACIÓN

1.4.1. JUSTIFICACIÓN TEÓRICA

El mercado actual de software es altamente competitivo y la tecnología es muy cambiante. Y en el desarrollo del software se pide básicamente rapidez, calidad y reducción de costes. Pero para asumir estos retos, es necesario tener agilidad y flexibilidad. Los ciclos de desarrollo por un lado, acostumbran a ser largos, y lo que se exige por otra parte, es que esos ciclos sean lo más cortos posibles.

Por ello, el presente estudio se respalda con los fundamentos teóricos de la Metodología Scrum, y se justifica su aplicación por permitir la agilidad y flexibilidad que se dan en un ciclo constante de reuniones diarias que involucra al recurso humano involucrado en el proceso de desarrollo de software.

1.4.2. JUSTIFICACIÓN PRÁCTICA

La siguiente investigación se justifica desde el punto de vista práctico porque permitirá agilizar los procesos permitiéndole al personal la migración de una tecnología a otra de manera ágil, incrementando así la productividad en el proceso de desarrollo de software de la unidad de negocio de TI para la empresa CCJ,

Con este trabajo se pretende dar a conocer la manera de poder implantar esta nueva metodología en las diversas organizaciones, habiendo precedentes de otras que obtuvieron buenos resultados después de aplicarla al contexto de su organización.

1.4.3. JUSTIFICACIÓN METODOLÓGICA

Existen varias metodologías de software, pero varias empresas peruanas, específicamente CCJ se está quedando atrás debido a que no innova sus procesos productivos.

Es por ello que, para poder entender cuál es la interrelación que existe entre los principales factores que influyen en el desarrollo de software y cuáles son los principales efectos, se hará uso de Scrum como metodología para poder optimizar y agilizar los procesos en la organización en estudio; y poder así dar luces a diversas empresas inmersas en el desarrollo de software para que estén a la vanguardia y puedan replicar Scrum en sus organizaciones.

1.5. HIPÓTESIS

1.5.1. HIPÓTESIS GENERAL

- La Metodología Scrum influye significativamente sobre el incremento de la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C.

1.5.2. OPERACIONALIZACIÓN DE LAS VARIABLES E INDICADORES DE LA HIPÓTESIS

VARIABLES	DEFINICIÓN	INDICADOR	UNIDAD DE MEDIDA	INSTRUMENTO DE EVALUACIÓN	FUENTES DE VERIFICACION
VARIABLE INDEPENDIENTE					
Metodología Scrum	Aplicación de la metodología Scrum en el proceso de desarrollo	Nivel de conocimiento o grado de dominio de la metodología	- Horas de capacitación	Aplicación práctica de la metodología a un caso	Certificados de capacitación en la metodología Scrum
VARIABLE DEPENDIENTE					
Productividad en el desarrollo de software	Cantidad de proyectos terminados en proporción con los recursos empleados	Porcentaje o índice de retraso en la entrega de proyectos	-Días de retraso	Cálculo del porcentaje desfase	Cumplimiento de fechas en cronogramas por proyecto

1.6. DISEÑO METODOLÓGICO

1.6.1. TIPO DE INVESTIGACIÓN

El tipo de investigación utilizada es la investigación aplicada, ya que se parte de los conocimientos adquiridos, además se hace uso de diversas fuentes, todas ellas referidas a la aplicación de la metodología Scrum en distintas organizaciones, que buscan optimizar sus procesos y por consecuencia incrementar su productividad.

1.6.2. NIVEL DE INVESTIGACIÓN

El nivel de la investigación de acuerdo al tipo de investigación es descriptivo, donde se empleará la metodología Scrum para poder determinar su implicancia y/o relación con la productividad del proceso de desarrollo de software.

El indicador de la productividad será medido antes y después de aplicar esta metodología a la unidad de negocio en estudio.

1.6.3. SISTEMA DE REFERENCIA

La investigación se aplica principalmente a la unidad de negocio de Tecnologías de Información de la Empresa CCJ S.A.C., ubicada en el distrito de Ate Vitarte en la ciudad de Lima.

Es una unidad de negocio relativamente pequeña y está compuesta de cinco colaboradores: A la cabeza la Gerencia de TI (01), el Líder del Equipo (01), el Gestor Comercial (01) y los Analistas Programadores de Sistemas (02).

En el presente estudio, se aplicaron encuestas y entrevistas al todo el personal que labora en la unidad de negocio de TI.

Como se pudo apreciar a lo largo de este primer capítulo Generalidades, se presentaron los principales problemas que presenta el proceso productivo de la unidad de negocio de TI de la empresa CCJ S.A.C., además se dio a conocer el problema formulado, el objetivo que se desea alcanzar, la justificación del porqué se realiza la presente investigación aplicando la metodología Scrum, la hipótesis que se pretende probar y el respectivo diseño metodológico.

CAPÍTULO II

MARCO DE REFERENCIA

El siguiente capítulo se enfocará en aspectos concernientes al marco de referencia, tales como antecedentes, con el propósito de obtener información de estudios anteriores que permitan ampliar el panorama para dar una adecuada solución al problema planteado en el capítulo anterior; así como también el marco teórico sobre las bases que fundamentan la Metodología Scrum; además del modelo aplicativo que permite aplicar paso a paso la metodología y como término el marco conceptual.

2.1. ANTECEDENTES

A.1. Rodríguez González, Pilar (2008). Estudio de la Aplicación de Metodologías Ágiles para la Evolución de productos software. Tesis de Máster para obtener el Título de Máster en Tecnologías de la Información. Facultad de Informática Universidad Politécnica de Madrid. Madrid.

Las actuales características de dinamismo y variabilidad de la industria del software han precisado replantear los cimientos sobre los que se sustenta el desarrollo software convencional. Un reciente estudio realizado por Boehm, sobre la tendencia en ingeniería del software, indica que el mercado actual está caracterizado por el rápido desarrollo de aplicaciones y la reducción de la vida de los productos. En este entorno inestable la ventaja competitiva se encuentra en aumentar la productividad y satisfacer las variantes necesidades del cliente en el menor tiempo posible para proporcionar un mayor valor al negocio. Ante esta situación, cabe reflexionar sobre el grado de adaptación de las metodologías convencionales a estas circunstancias. La mayoría de los estudios coinciden en que el carácter normativo y la fuerte dependencia de planificaciones previas al desarrollo que definen a las metodologías convencionales, implican que resulten

excesivamente pesadas para cubrir las necesidades de un amplio porcentaje del mercado software actual.

En los últimos años las metodologías ágiles han irrumpido con fuerza como un intento de despojar al desarrollo software del estricto corsé planteado por las metodologías convencionales, y son muchas las organizaciones punteras con creciente interés en las mismas. La novedad de estas metodologías hace que, aunque existen evidencias de los beneficios que pueden proporcionar en proyectos de pequeña envergadura, aun resulte difícil escalar a grandes proyectos. Algunos estudios recientes indican que la productividad y calidad del software aumenta aplicando los principios y valores que las rigen. No obstante, la mayoría de estos estudios se limitan a narrar observaciones cualitativas. Entre los que utilizan datos empíricos para apoyar sus conclusiones, los resultados son tan dispares como una mejora del 337% en la productividad y un decremento del 44%. Por este motivo, desde las organizaciones que promueven el desarrollo ágil de aplicaciones se solicita la realización de estudios sobre metodologías ágiles que permitan constatar o reprobar sus beneficios.

El objeto de esta investigación es estudiar la evolución de un producto de software concreto utilizando las directrices marcadas por metodologías ágiles, en concreto por la metodología SCRUM. Se presentan los resultados obtenidos en aspectos tales como las características del producto a lo largo de la evolución, incluyendo estimaciones de la calidad del producto obtenido, la agilidad en el desarrollo, y evaluando el esfuerzo dedicado a adoptar la metodología. Además, dado que el factor humano es fundamental en este tipo de metodologías, se presenta un análisis cualitativo del desarrollo del proyecto.

Cabe destacar que el estudio aquí presentado se enmarca en una de las líneas de investigación del grupo SYST (System and Software Technology Group) de la Universidad Politécnica de Madrid, que participa en el proyecto ITEA2 Flexi. En este proyecto se persigue mejorar la competitividad de la industria software europea proporcionando un entorno flexible, rápido y ágil para el desarrollo de aplicaciones que permita adaptarse a las actuales características del mercado para pasar de la idea al producto en seis meses.

La referida tesis aporta al presente trabajo de investigación, debido a que tiene como objeto el estudio de la evolución de un producto de software para lo cual hace uso de las directrices marcadas por metodologías ágiles, específicamente la metodología SCRUM. También incluye estimaciones de la calidad del producto obtenido, la agilidad en el desarrollo, y evalúa el esfuerzo dedicado a

adoptar la metodología. Además, dado que el factor humano es fundamental en este tipo de metodologías, se presenta un análisis cualitativo del desarrollo del proyecto.

A.2. Nakashima Chávez, Giancarlo Juan (2009). Mejora del Proceso de Software de una Empresa Desarrolladora de Software: Caso COMPETISOFT – PERÚ DELTA. Tesis para optar por el Título de Ingeniero Informático. Pontificia Universidad Católica del Perú Facultad de Ciencias e Ingeniería. Lima.

El mencionado proyecto de fin de carrera corresponde a la ejecución de un ciclo de mejora de procesos realizado en una pyme que se dedica principalmente al desarrollo de software, denominada en el proyecto y en el presente documento, por acuerdo de confidencialidad como DELTA.

Para la realización del ciclo de mejora se utilizó el modelo COMPETISOFT, “Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica”, financiado por la CYTED (Ciencia y Tecnología para el Desarrollo). Es importante resaltar que este modelo está siendo implantado y probado en diversas empresas peruanas desarrolladoras de software, a través de un trabajo conjunto con alumnos de la Pontificia Universidad Católica del Perú, grupo denominado COMPETISOFT-PUCP.

El presente trabajo consta de 4 capítulos. En el primero se describen algunos modelos de calidad que actualmente se aplican, luego se detalla sobre el modelo COMPETISOFT y por último se analiza la situación actual de las empresas desarrolladoras de software en el Perú. En el capítulo dos se presenta una breve descripción de la empresa Delta. En la segunda sección se muestran los resultados obtenidos luego de una evaluación inicial de todos los procesos, indicando cuales alcanzaron el nivel 1 dentro del modelo COMPETISOFT. Para finalizar este capítulo, se detalla el esquema de trabajo utilizado en el presente proyecto. En el tercero se presentan los procesos seleccionados para realizar la mejora, así como también cuales procesos van a ser afectados indirectamente. Luego para cada proceso a mejorar, se muestra en un diagrama la situación inicial, la propuesta de mejora y se comentan las experiencias acerca del ciclo de mejora. Por último, se presenta la evaluación final de todos los procesos, se calcula el esfuerzo del proyecto y se sugieren directrices para un nuevo ciclo de mejora. Finalmente, en el capítulo 4 se describen las observaciones, conclusiones y recomendaciones que puedan ser útiles para un siguiente ciclo de mejora de procesos.

El mencionado estudio contribuye a la presente tesis debido a que muestra un ejercicio práctico de la ejecución de un ciclo de mejora de procesos realizado en una pyme que se dedica principalmente al desarrollo de software, utilizando el modelo Competisoft. Modelo que viene siendo implantado y probado en diversas empresas peruanas desarrolladoras de software.

A.3. Mudarra Teruel, Héctor y Pons Aróztegui, Jordi (2010). Automatización de Sistemas de Desarrollo Ágil – SCRUM: TEAM & ROLE. Memoria del Proyecto de Fin de Carrera de Ingeniería Informática. Universidad Autónoma de Barcelona. Bellaterra.

Cuando se quiere desarrollar un proyecto software primero hay que tener claro qué es lo que se quiere obtener y luego preguntarse cómo hacerlo. El cómo acostumbra a ser siempre una decisión por parte de los analistas o bien de los programadores, aunque en raras excepciones se pide una forma concreta de hacer las cosas. Sin embargo el qué no depende del equipo sino del cliente que pide la aplicación, y para transmitir ese qué debe de hacerlo con palabras, esquemas o dibujos. Una vez se conoce el qué y se ha tomado una decisión sobre el cómo, se entra en la fase de desarrollo donde se pasa un tiempo hasta obtener un resultado funcional.

El modelo clásico de desarrollo de un proyecto software es el modelo en cascada. En este modelo, una vez han quedado debidamente definidos todos los requisitos, se realiza una división en tareas y se les asigna una prioridad. Entonces se hace una planificación temporal con estas tareas y se empieza el desarrollo siguiendo lo especificado. Si todo va según lo planeado el proyecto terminará en el tiempo previsto. Si no ha ido según lo planeado cuando llegue la fecha final habrán quedado pendientes aquellas tareas con una menor prioridad. Por tanto, usando el modelo clásico, se habrá obtenido un resultado al cabo de un tiempo de desarrollo y se presentará lo realizado al cliente. Generalmente es en este momento donde surgen problemas ya que el cliente puede no quedar del todo satisfecho con el proyecto. Esto es debido a que resulta muy difícil transmitir exactamente lo que uno piensa y probablemente en la definición de requerimientos que se hizo tiempo atrás no logró plasmar lo que se pretendía. Por otro lado también es posible que el cliente haya decidido cambiar de idea en algunos aspectos a mitad de desarrollo y sus nuevos conceptos no concuerden con los resultados. Sea como sea, el resultado final no será del todo satisfactorio para el cliente y es muy probable que haya que hacer modificaciones que

ocasionen pérdida de mucho tiempo invertido en el desarrollo debido a cambios en los requerimientos.

Analizando el modelo clásico se encuentran principalmente dos puntos que resultan conflictivos: El primero es la dificultad de una especificación de requerimientos que sea concorde con lo que quiere el cliente. Como ya se ha comentado anteriormente aunque se haga exactamente lo que se pretendía es probable que se quieran hacer cosas nuevas o se haya cambiado de idea a mitad del desarrollo. Al no ser un conflicto que depende del equipo de desarrollo se desestima la búsqueda de solución por esta vía. El segundo es el tiempo que transcurre entre la definición de requerimientos y la muestra de resultados. Este período de tiempo está directamente relacionado con la magnitud del proyecto a desarrollar, sin embargo aquellos proyectos con un mayor tiempo acostumbran a ser también aquellos que tienen más inversión y donde los costes de un mal resultado son mayores. En este caso sí que el equipo de desarrollo tiene control y se puede buscar una solución.

Una solución que ha nacido para combatir los largos tiempos de espera entre la definición y el resultado y la poca flexibilidad de cambio que tienen los proyectos son las metodologías de desarrollo de software ágiles. En este proyecto se estudia y se hace uso de una, en concreto el método *Scrum*.

El referido proyecto aporta a la presente tesis gracias a que en él se estudia la evolución de la implementación de un proyecto desarrollado mediante la metodología Scrum y el autor realiza el respectivo seguimiento. El proyecto desarrollado consta de cuatro módulos que, en su conjunto, constituyen una aplicación capaz de gestionar los recursos de la metodología.

A.4. Matthews Schele, Wesley David (2005). Adaptation of a software development methodology incorporating Social Network Analysis and Situational Leadership. Tesis para optar por el Título de Ingeniero Civil en Informática. Universidad Austral de Chile Facultad de Ciencias de la Ingeniería. Valdivia.

Este trabajo es un estudio de metodologías de desarrollo de software y de cómo pueden ser optimizadas mediante la incorporación de métodos analíticos para el diagnóstico de situaciones. Al incorporar SNA (Análisis de Redes Sociales) y Liderazgo Situacional, trata de resolver problemas comunes a todas las metodologías.

El trabajo primero establece la situación general de del desarrollo de software y luego analiza las metodologías más populares disponibles. De la lista que se discute se selecciona SCRUM para su modificación. Luego de analizar SNA y el liderazgo situacional, se propone una modificación a la metodología.

La metodología se descompone en tres etapas, definición de requerimientos, desarrollo e implantación. Luego la metodología es modificada utilizando técnicas de SNA y se propone un modelo para un método cuantitativo para la evaluación de los niveles de preparación de liderazgo situacional. La nueva metodología modificada se denomina Social Scrum o S-Scrum. A continuación la metodología se prueba en un proyecto real. Finalmente se discuten las conclusiones referentes a la nueva metodología y a los modelos analíticos propuestos.

La metodología resultante tiene por objetivo ser de utilidad como una herramienta de evaluación y recomendación para jefes de proyecto y arquitectos de solución. Al proveer información adicional y métodos de control en varios puntos clave en la metodología de desarrollo seleccionada, se busca un mejor proceso, y una posible ruta a encontrarlo queda propuesta.

El mencionado trabajo de investigación contribuye a la presente tesis porque es un estudio de metodologías de desarrollo de software y de cómo éstas pueden ser optimizadas mediante la incorporación de métodos analíticos para el diagnóstico de situaciones. Por ejemplo; al incorporar SNA (Análisis de Redes Sociales) y Liderazgo Situacional, trata de resolver problemas comunes a todas las metodologías.

A.5. Spada, Danilo (S.A.D.P.1). Usabilidad en el proceso de desarrollo de SCRUM. Tesis para optar por el Grado en Máster en Ingeniería Informática y de Telecomunicación. Programa Oficial de Postgrado en Ingeniería Informática y de Telecomunicación. México D.F.

Según el ISO/IEC 9126, la calidad de un producto software depende de seis factores: funcionalidad, fiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad. Sin embargo, aunque la mayoría de estos factores se tienen en cuenta a la hora de plantear el desarrollo de un sistema o software, la usabilidad muchas veces no tiene un papel adecuado. Pero, ¿qué se entiende por usabilidad de un sistema software? Pues es la medida de cuanto el sistema sea

¹ Sin año de publicación

sencillo de aprender y utilizar, de la baja probabilidad de cometer errores durante su uso, incluyendo además la satisfacción y la eficiencia de los usuarios finales. Durante los últimos años, sobre todo debido a la gran afluencia de una multitud de usuarios con diferentes perfiles a muchas aplicaciones a través Internet, el tema de la usabilidad está cobrando una importancia cada día más relevante. La usabilidad determina cuan eficientemente y cómodamente un usuario alcanza su objetivo. Si nos fijamos que los sistemas de software a menudo pretenden intervenir en la realización de alguna tarea más o menos compleja, ayudando y optimizando el esfuerzo del usuario, pues es indispensable que éste pueda aprovechar la ayuda sin sufrir. Interfaces demasiado complejas o procedimientos largos y poco intuitivos, comportamientos incomprensibles del sistema, etc. pueden volver estresante y agobiante tareas que deberían resultar más sencillas y controladas.

En el desarrollo de una nueva aplicación es imprescindible tener en cuenta que un producto, hasta muy sofisticado, pero no usable, puede ser un fracaso total; aquí vale la regla: si el usuario no encuentra o no entiende una funcionalidad, ¡la funcionalidad no existe! Aún más: puede que sea necesario afrontar un coste muy alto para arreglar faltas en usabilidad: se tome como ejemplo lo que pasó en EEUU en el 2000 cuando hubo que resolver el problema de la dificultad de comprensión y utilización de papeletas electoral para votar el nuevo presidente.

Craig Larman sostiene que "no hay probablemente ninguna otra técnica con mayor disparidad entre su importancia para el éxito del desarrollo de software y la falta de una atención y educación formal que la ingeniería de usabilidad y el diseño de las interfaces de usuario". Muchos esfuerzos se están haciendo para demostrar la oportunidad de invertir en usabilidad y hay trabajos que muestran como para cada dólar invertido en usabilidad, se obtienen de \$10 hasta \$100.

Está claro entonces, que los ingenieros deberían tener más en cuenta la usabilidad en sus proyectos, y eso durante todas las fases del desarrollo, desde el diseño hasta la evaluación. Sin embargo en los estándares más utilizados para desarrollar proyectos software (IEEE Std 1074-1997, Modelo en cascada, Modelo incremental, Extreme Programming, SCRUM) no está muy claro qué medidas tomar para mejorar este aspecto a lo largo de todo el ciclo de vida y hasta ahora no han sido añadido "etapas" vueltas precisamente a obtener productos usables. De alguna manera se puede pensar a este problema como a una independencia todavía "impermeable" entre IPO (Interacción Persona-Ordenador) e IS (Ingeniería del software).

Sin embargo la industria del software está creciendo cada día más y las técnicas IPO empiezan a ser maduras y está naciendo la necesidad de integrar los conocimientos de estos dos campos para que los productos finales, los sistemas software, sus usuarios, pero también los ingenieros que los desarrollan, puedan aprovechar los conocimientos madurados. El profesor Xavier Ferrè en su tesis doctoral nos ofrece un resultado importante en este sentido: nos indica una metodología simple, clara y flexible para integrar las técnicas de usabilidad en un cualquier proceso de desarrollo software incremental. Como él mismo dice, "el único requisito que debe cumplir un proceso para poder integrar técnicas de usabilidad con dicho objetivo es estar basado en un enfoque iterativo, pues tal característica es clave para desarrollar productos con un buen nivel de usabilidad".

El referido estudio aporta a la presente tesis el hecho de que pretende ofrecer la descripción de un proceso de desarrollo ágil Scrum y aplicar a este proceso el Marco de Integración de la Usabilidad desarrollado por el profesor Xavier Ferré en su tesis doctoral. Se presentan entonces las técnicas y las actividades de IPO elegidas y se indican los momentos de aplicación en un posible caso concreto.

2.2. MARCO TEÓRICO

2.2.1. INTRODUCCIÓN AL MODELO ÁGIL DE DESARROLLO DE SOFTWARE

Las metodologías ágiles son sin duda uno de los temas recientes en ingeniería de software que están acaparando gran interés y controversia. A mediados de los años 90 comenzó a forjarse una definición moderna de desarrollo ágil del software como una reacción contra las metodologías utilizadas hasta el momento, consideradas excesivamente pesadas y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. En el año 2001, diecisiete miembros destacados de la comunidad de software, incluyendo algunos de los creadores o impulsores de las metodologías en software, se reunieron en Utah (Estados Unidos) y adoptaron el nombre de "Metodologías ágiles" para denominar a esta nueva corriente de desarrollo. Poco después, algunos de estos miembros formaron la conocida como "Alianza ágil", una organización sin ánimo de lucro que promueve el desarrollo ágil de aplicaciones. Desde ese momento hasta la actualidad las metodologías ágiles han ido adquiriendo gran auge dentro de la industria del software y las organizaciones líderes comienzan a apostar por este nuevo enfoque para desarrollar sus productos.

A. NECESIDAD DE USAR METODOLOGÍA ÁGILES

Tradicionalmente se ha tendido a desarrollar software a través de metodologías que encajaban en el proceso de desarrollo de manera un tanto rígida, cada vez más, se demuestra que esto es erróneo por las actuales características de dinamismo y variabilidad del mercado software. Se tiende hacia el rápido desarrollo de aplicaciones y la vida de los productos se acorta. En este entorno inestable, que tiene como factor inherente el cambio y la evolución rápida y continua, la ventaja competitiva se encuentra en aumentar la productividad y satisfacer las variantes necesidades del cliente en el menor tiempo posible para proporcionar un mayor valor al negocio.

Sin embargo, las metodologías convencionales presentan diversos problemas a la hora de abordar un amplio rango de proyectos industriales en este turbulento entorno. Entre estos problemas podemos destacar los siguientes:

Perciben la captura de requisitos del proyecto como una fase previa al desarrollo del mismo que, una vez completada, debe proporcionar una fotografía exacta de qué desea el cliente. Se trata de evitar a toda costa que se produzcan cambios en el conjunto de requisitos inicial, puesto que a medida que avanza el proyecto resulta más costoso solucionar los errores detectados o introducir modificaciones, y pretenden delegar toda responsabilidad económica en el cliente en caso de que estos cambios de requisitos se produzcan. Por este motivo, se les conoce también como metodologías predictivas. Sin embargo, el esfuerzo, tanto en coste como en tiempo, que supone hacer una captura detallada de todos los requisitos de un proyecto al comienzo del mismo es enorme y rara vez se ve justificado con el resultado obtenido. Además, en muchas ocasiones el cliente no conoce sus propias necesidades con la profundidad suficiente como para definir las de forma exacta a priori y, a menudo, estas necesidades y sus prioridades varían durante la vida del proyecto. Establecer mecanismos de control es una de las opciones existentes para protegerse de estos cambios, aunque frecuentemente provocan la insatisfacción de los clientes, que perciben el desarrollo del proyecto como algo inflexible que no

se adapta a sus necesidades y que si lo hace repercute negativamente en costes añadidos al presupuesto del proyecto.

Por otro lado, en muchas ocasiones el proceso de desarrollo convencional está oprimido por excesiva documentación no siempre útil. Un porcentaje elevado del tiempo de desarrollo de un producto software se dedica o, desde el punto de vista de las metodologías ágiles, se malgasta en crear documentación que finalmente no se utiliza y que, por tanto, no aporta valor al negocio. Además, esta documentación innecesaria entorpece las labores de mantenimiento de la propia documentación útil lo que provoca que en muchas ocasiones el mantenimiento de la documentación se obvie agudizando, de este modo, el coste en las tareas de documentación futuras. Evidentemente, estas circunstancias no se adaptan a las restricciones de tiempo del mercado actual.

Otra dificultad añadida al uso de metodologías convencionales es la lentitud del proceso de desarrollo. Es difícil para los desarrolladores entender un sistema complejo en su globalidad lo que provoca que las diferentes etapas del ciclo de vida convencional transcurran lentamente. Dividir el trabajo en módulos abordables ayuda a minimizar los fallos y, por tanto, el coste de desarrollo. Además, permite liberar funcionalidad progresivamente. En la feroz competencia del mercado vigente, en la que los productos quedan obsoletos rápidamente, se pide básicamente rapidez, calidad y reducción de costes, pero para asumir estos retos, es necesario tener agilidad y flexibilidad.

Asimismo, las metodologías convencionales tienden a acumular los riesgos y dificultades que surgen en el desarrollo del producto al final del proyecto, repercutiendo en retrasos en la entrega de productos o influyendo en la incorrecta ejecución de las últimas fases del ciclo de vida.

B. VENTAJAS DE USAR METODOLOGÍAS ÁGILES

En contraposición a las metodologías convencionales, las metodologías ágiles aparecen como alternativa atractiva para adaptarse a entornos de baja productividad en el desarrollo de software, debido a que son apropiadas cuando los requisitos son emergentes y cambian rápidamente. De este modo, presentan diversas ventajas en el contexto actual:

- ✓ Capacidad de respuesta a cambios a lo largo del desarrollo ya que no se perciben como un impedimento sino como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente,

considerando la gestión de cambios como un aspecto característico del propio proceso de desarrollo software.

- ✓ Entrega continua y en plazos breves de software funcional lo que permite al cliente verificar in situ el desarrollo del proyecto, ir disfrutando de la funcionalidad del producto progresivamente y comprobando si satisface sus necesidades, mejorando de esta forma su satisfacción. Además, el desarrollo en ciclos de corta duración favorece que los riesgos y dificultades se repartan a lo largo del desarrollo del producto, principalmente al comienzo del mismo y permite ir aprendiendo de estos riesgos y dificultades.
- ✓ Trabajo conjunto entre el cliente y el equipo de desarrollo con una comunicación directa que pretende mitigar malentendidos, que constituyen una de las principales fuentes de errores en productos de software, y exceso de documentación improductiva.
- ✓ Importancia de la simplicidad, eliminando el trabajo innecesario que no aporta valor al negocio.
- ✓ Atención continua a la excelencia técnica y al buen diseño para mantener una alta calidad de los productos.
- ✓ Mejora continua de los procesos y del equipo de desarrollo, entendiendo que el éxito, depende de tres factores: éxito técnico, éxito personal y éxito organizacional.

C. EL MANIFIESTO ÁGIL: VALORES Y PRINCIPIOS

En este punto se profundizará en los valores y principios que sustentan las metodologías ágiles. Continuamente se utiliza el concepto de agilidad para definir estas metodologías, pero ¿qué significa ser ágil desde una perspectiva de software? Basados en el consenso de varias definiciones contemporáneas, se obtuvo la siguiente: *“La agilidad es un comportamiento persistente o habilidad, de entidad sensible, que presenta flexibilidad para adaptarse a cambios, esperados o inesperados, rápidamente; persigue la duración más corta en tiempo; usa instrumentos económicos, simples y de calidad en un ambiente dinámico; y utiliza los conocimientos y experiencia previos para aprender tanto del entorno interno como del externo.”*²

² Rodríguez, G. P, (2008). *Estudio de la Aplicación de Metodologías Ágiles para la evolución de productos software. Tesis de Máster en Tecnologías de la Información, Facultad de Informática. Madrid.*

Por tanto, el desarrollo ágil no especifica unos procesos o métodos que seguir, aunque bien es cierto que han aparecido algunas prácticas asociadas a este movimiento. El desarrollo ágil es más bien una filosofía de desarrollo software.

En el gráfico N° 2.1 se plasma el Manifiesto para el desarrollo ágil de software, que es el punto de partida donde se establecen las ideas emanadas tras la reunión de Utah.

Gráfico N° 2. 1
Manifiesto para el desarrollo ágil de software



Fuente: http://www.scrummanager.net/files/sm_proyecto.pdf

Elaboración: Palacio Juan. Scrum Manager Gestión de proyectos

Del gráfico, se observa que el Manifiesto ágil es un documento que resume la filosofía agile estableciendo cuatro valores y doce principios. Según el Manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proceso software. Este primer valor expresa que es preferible utilizar un proceso indocumentado con buenas interacciones personales que un proceso documentado con interacciones hostiles. Se considera que no se debe pretender construir primero el entorno y esperar que el equipo se adapte automáticamente sino al revés, construir primero el equipo y que éste configure su propio entorno. El talento, la habilidad, la capacidad de comunicación y de tratar con

personas son características fundamentales para los miembros de un equipo ágil.

- Desarrollar software que funcione por encima de una completa documentación. Este valor es utilizado por muchos detractores de las metodologías ágiles que argumentan que éstas son la excusa perfecta para aquellos que pretenden evitar las tareas menos gratificantes del desarrollo de software como las tareas de documentación. Sin embargo, el propósito de este valor es acentuar la supremacía del producto por encima de la documentación. El objetivo de todo desarrollador es obtener un producto que funcione y cumpla las necesidades del cliente y la documentación es un artefacto que se utiliza para cumplir su objetivo. Por tanto, no se trata de no documentar sino de documentar aquello que sea necesario para tomar de forma inmediata una decisión importante. Los documentos deben ser cortos y centrarse en lo fundamental. Dado que el código es el valor principal que se obtiene del desarrollo se enfatiza en seguir ciertos estándares de programación para mantener el código legible y documentado.
- La colaboración con el cliente por encima de la negociación contractual. Se propone una interacción continua entre el cliente y el equipo de desarrollo de tal forma que el cliente forme un tándem con el equipo de desarrollo. Se pretende no diferenciar entre las figuras cliente y equipo de desarrollo sino que se apuesta por un solo equipo persiguiendo un objetivo común.
- Responder a los cambios más que seguir estrictamente un plan. Planificar el trabajo a realizar es muy útil y las metodologías ágiles consideran actividades específicas de planificación a corto plazo. No obstante, adaptarse a los cambios es vital en la industria software actual y, por tanto, también consideran mecanismos para tratar los cambios de prioridades. La regla es: "Planificar es útil. Seguir un plan es útil hasta que el plan se distancia de la situación actual. Pender de un plan desactualizado es caminar por un callejón sin salida".

Para cumplir estos valores se siguen doce principios que establecen algunas diferencias entre un desarrollo ágil y uno convencional:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporten valor.

2. Dar la bienvenida a los cambios de requisitos. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Liberar software que funcione frecuentemente, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. Los miembros del negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y apoyo que necesiten y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la principal medida de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos que se auto organizan.
12. En intervalos regulares el equipo debe reflexionar sobre cómo ser más efectivo y según estas reflexiones ajustar su comportamiento.

Estos principios marcan el ciclo de vida de un desarrollo ágil, así como las prácticas y procesos a utilizar, siendo que la forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la comunicación cara a cara, tal como lo especifica el Manifiesto ágil.

2.2.2. CÓMO SER ÁGIL. ALGUNAS PRÁCTICAS

Aunque el movimiento ágil está sustentado por valores y principios para el desarrollo de productos de software, la mayoría de estas metodologías tienen asociadas un conjunto de prácticas, en muchos casos comunes, que buscan la agilidad en el desarrollo. En esta sección vamos a analizar cinco de las prácticas más relevantes:

A. Primera práctica: Planificación, Historias de usuario.

Mientras que las metodologías convencionales centran la ingeniería de requisitos en habilidades de predicción basándose en férreas planificaciones previas al desarrollo, las metodologías ágiles perciben la gestión de cambios como un aspecto inherente al proceso de desarrollo software, considerando planificaciones continuas, más flexibles y en cortos plazos, que respondan mejor a los cambios en las necesidades del cliente.

Con el objetivo de disminuir el coste que supone la etapa de planificación en las metodologías convencionales, las ágiles simplifican las tareas de gestión y documentación de las necesidades del sistema. Apuestan por involucrar de una forma más intensa al cliente a lo largo de todo el proceso de desarrollo, de forma que, la comunicación directa y frecuente retroalimentación son las prácticas más importantes para la planificación y especificación de requisitos en estas metodologías.

En este proceso de definición de las necesidades del sistema que guiará el desarrollo, se utilizan las llamadas historias de usuario para detallar, en un lenguaje cercano al cliente, la funcionalidad que debe satisfacer el sistema. Las historias de usuario se descomponen en tareas que deberán ser realizadas para cumplir con la funcionalidad que se solicita. Se debe notar que la estimación de tiempo no es una restricción fija, simplemente constituye una aproximación inicial. Se considera que las historias de usuario deben cumplir seis características: ser independientes, negociables, valorables, estimables, pequeñas y comprobables.

B. Segunda práctica: Programación en pareja (Pair Programming).

Una de las prácticas más utilizadas en metodologías ágiles, sobre todo en sistemas de alta complejidad, es la programación en parejas, que establece que la producción de código se realice con trabajo en parejas de programadores. El utilizar Pair Programming en un proyecto no implica que todas las líneas de código sean escritas por una pareja, ya que mientras una persona está escribiendo el código, la otra puede estar verificando errores, pensando en alternativas mejores, identificando casos de prueba, pensando en aspectos de diseño, etc. El objetivo principal de esta técnica es disminuir la tasa de errores, mejorar el diseño y aspectos como la satisfacción de los programadores. No obstante, algunos estudios indican que se trata de una técnica intensa y estresante para los propios programadores, aunque admiten que acelera el proceso de desarrollo.

Otro detalle que se debe tener en cuenta son las habilidades y capacidades de los miembros de la pareja. Si existe una diferencia importante entre ellos, puede llegar a ser una práctica frustrante para ambos. En este caso concreto, el Pair Programming ha de ser visto como una técnica de aprendizaje.

C. Tercera práctica: Integración continua (Continuous integration).

La integración continua pretende mitigar la complejidad que el proceso de integración suele tener asociada en las metodologías convencionales, superando, en determinadas circunstancias, la propia complejidad de la codificación. El objetivo es integrar cada cambio introducido, de tal forma que pequeñas piezas de código sean integradas de forma continua, ya que se parte de la base de que cuanto más se espere para integrar más costoso e impredecible se vuelve el proceso. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

Para que esta práctica sea viable es imprescindible disponer de una batería de test, preferiblemente automatizada, de tal forma, que una vez que el nuevo código esté integrado con el resto del sistema se ejecute toda la batería de pruebas. Si son pasados todos los test del sistema, se procede a la siguiente tarea. En caso contrario, aunque no falle el nuevo módulo que se quiere introducir en el sistema, se ha de regresar a la versión anterior, pues ésta ya había pasado la batería de pruebas.

D. Cuarta práctica: Refactorización.

Actividad constante en los desarrollos ágiles cuyo objetivo es mejorar el diseño de un sistema sin influir en su funcionalidad. Se pretende reestructurar el código para eliminar duplicaciones, mejorar su legibilidad, simplificarlo y hacerlo más flexible de forma que se faciliten los posteriores cambios. Recordemos que el código que funciona es el principal aporte de valor para las metodologías ágiles, por encima de la documentación, por lo que se enfatiza en que éste sea legible y permita la comunicación entre desarrolladores.

E. Quinta práctica: Desarrollo dirigido por pruebas (Test Driven Development).

Al utilizar esta práctica la producción de código está dirigida por las pruebas. Concretamente, se trata de escribir las pruebas que validarán el sistema antes de comenzar a construirlo, idealmente que sea el propio cliente quien las escriba. De esta forma, ante cada modificación del sistema la batería completa de pruebas es ejecutada. Esta práctica

constituye, por tanto, la primera línea para garantizar la calidad del producto, pues ésta se considera en términos de la correcta funcionalidad que se le ofrece al cliente. Entre las ventajas que aporta el desarrollo dirigido por pruebas podemos destacar que disminuye el tiempo dedicado a solucionar errores y genera un sentimiento de confianza de éxito del proyecto en el equipo.

No obstante, el hecho de que aspectos tan importantes como la cohesión o el acoplamiento de los módulos pase a un segundo plano en el desarrollo a favor de la funcionalidad, implica la necesidad de realizar constantes etapas de refactorización que permitan mantener la calidad del código.

Como se puede apreciar, a pesar de la reciente aplicación de las metodologías ágiles la mayoría de las prácticas propuestas no son novedosas sino que de alguna manera ya habían sido propuestas en ingeniería de software. El mérito de las metodologías ágiles es proponer una aplicación conjunta, equilibrada y efectiva de forma que se complementen con ideas desde la perspectiva del negocio, los valores humanos y el trabajo.

2.2.3. REVISIÓN DE OTRAS METODOLOGÍAS ÁGILES

Aunque el manifiesto ágil es la piedra angular sobre la que cimientan todas las metodologías ágiles, cada una tiene sus propias características. En esta sección se describen, a grandes rasgos, las particularidades fundamentales de algunas otras metodologías ágiles, que actualmente tienen gran aceptación en el mercado.

A. eXtreme Programming, también conocida como XP, es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Aspectos como el trabajo en equipo, el aprendizaje de los desarrolladores y propiciar un buen clima de trabajo son pilares en esta metodología. Oficialmente fue creada en 1999 por Kent Beck, con la publicación de su libro *Extreme Programming Explained*.

XP se centra en la continua retroalimentación entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como una metodología especialmente adecuada para proyectos con requisitos muy cambiantes e imprecisos, donde existe un

alto riesgo técnico. Como metodología pragmática, recoge las que considera mejores prácticas para el desarrollo software, cuya aplicación disciplinada pretende disminuir la curva exponencial del costo del cambio a lo largo del proyecto. Se trata de doce prácticas: el juego de la planificación, entregas pequeñas, metáfora, diseño simple, pruebas, refactorización, programación en parejas, propiedad colectiva del código, integración continua, 40 horas por semana, cliente in-situ y estándares de programación.

Una posterior revisión de la metodología clasificó las prácticas en primarias, aquellas que según Beck proporcionan una mejora inmediata independientemente de la metodología que se esté siguiendo, y secundarias, que implican una mayor dificultad si no se tiene experiencia en las prácticas primarias. Siguiendo esta clasificación, las prácticas primarias consideradas fueron la adecuación del lugar de trabajo, sentarse juntos, entorno de trabajo informativo, sentimiento de equipo, trabajo enérgico, programación en parejas, user stories, iteraciones cortas, integración continua, pruebas primero, diseño incremental y refactorización. El conjunto de prácticas secundarias quedó compuesto por la involucración del cliente, continuidad del equipo, equipos retractiles, código compartido y alcance del contrato negociado.

B. Crystal Methodologies, es un conjunto de metodologías ágiles para equipos de diferentes tamaños y con distintas características de criticidad. Fue propulsada por uno de los padres del Manifiesto Ágil, Alistair Cockburn, que consideraba que la metodología debe adaptarse a las personas que componen el equipo utilizando políticas diferentes para equipos diferentes. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores: Crystal Clear (3 a 8 miembros), Crystal Yellow (10 a 20 miembros), Crystal Orange (25 a 50 miembros), Crystal Red (50 a 100 miembros) y Crystal Blue (para más de 100 miembros). Por ejemplo, Crystal Clear, la metodología más ligera de este conjunto, está dirigida a la comunicación de equipos pequeños que desarrollan software cuya criticidad no es elevada. Tiene asociadas siete características: liberación frecuente de funcionalidad, mejora reflexiva, comunicación osmótica, seguridad personal, atención, fácil acceso para usuario expertos y requisitos para el entorno técnico.

- C. Dynamic Software Development Method (DSDM)**, puede considerarse un marco para el proceso de producción de software, más que una metodología. Nació en 1994 con el objetivo de crear una metodología RAD (Rapid Application Development) unificada. Divide el proyecto en tres fases: pre-proyecto, ciclo de vida del proyecto y post-proyecto, especificando de forma rigurosa la arquitectura y gestión del proyecto. Así, propone cinco fases en el desarrollo del proyecto: estudio de la viabilidad y estudio del negocio que constituyen la etapa de pre-proyecto; y, de forma iterativa, modelado funcional, diseño y construcción y finalmente implementación, además de una adecuada retroalimentación a todas las fases. Sus principales características son interacción con el usuario, poder del equipo de desarrollo, liberaciones frecuentes de funcionalidad, registrarse siguiendo las necesidades del negocio, desarrollo iterativo e incremental, adaptación a los cambios reversibles, fijar el nivel de alcance al comienzo del proyecto, pruebas durante todo el desarrollo y eficiente y efectiva comunicación.
- D. Feature-driven Development**, esta metodología, ideada por Jeff De Luca y Peter Coad, combina el desarrollo dirigido por modelos con el desarrollo ágil. Se centra en el diseño de un modelo inicial, cuyo desarrollo será dividido en función a las características que debe cumplir el software e, iterativamente, se diseñará cada una de estas características. Por tanto, cada iteración consta de dos partes, diseño e implementación de cada característica. Este tipo de metodología está dirigido al desarrollo de aplicaciones con un alto grado de criticidad.
- E. Lean Software Development**, es una metodología dirigida especialmente al desarrollo de sistemas cuyas características varían constantemente. Fue definida por Bob Charette's a partir de su experiencia en proyecto industriales, constituyendo una adaptación para el desarrollo de software de las lecciones aprendidas en la industria, en particular, en el sistema de producción automovilista japonesa de Toyota. La metodología establece que todo cambio en el desarrollo de software conlleva riesgos, pero si se manejan adecuadamente pueden convertirse en oportunidades que mejoren la productividad del cliente. Consta de siete principios dirigidos a gestionar los cambios: eliminación de todo aquello que no aporte valor al negocio, conocimiento incremental, toma de decisiones tan tarde como sea posible, deliberar funcionalidad tan pronto como sea posible, poder

del equipo, construcción incremental del producto y perspectiva global del proyecto.

2.2.4. ORIGEN DE LA METODOLOGÍA SCRUM

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre a principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujiro Nonaka a mediados de los 80. Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

Jeff Sutherland aplicó el modelo Scrum al desarrollo de software en 1993 en EaselCorporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation). En 1996 lo presentó junto con Ken Schwaber como proceso formal, también para gestión del desarrollo de software en OOPSLA 96. Más tarde, en 2001 serían dos de los promulgadores del Manifiesto Ágil. En el desarrollo de software Scrum está considerado como modelo ágil por la Agile Alliance.³

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

2.2.5. ¿POR QUÉ APLICAR SCRUM?

Siendo la presente tesis, aplicada a un proyecto real, con una fecha de inicio y de final, que se llevará a cabo de forma gradual, siguiendo una serie de pasos para obtener un producto, servicio o resultado único; es necesario elegir una adecuada metodología de gestión que permita dirigir el proyecto desde su inicio hasta su final de forma exitosa. Es por ello, que para el presente proyecto de tesis se ha decidido hacer uso de la metodología de gestión Scrum, metodología de gestión ágil que toma como base varios principios establecidos por el PMI.

A. VENTAJAS DE SCRUM

Dentro de las ventajas que ofrece la metodología Scrum, tenemos:

³ <http://www.navegapolis.net> – Juan Palacio, 2006

- Adecuado manejo de los requerimientos cambiantes
- Incentiva la motivación del equipo de desarrollo
- El cliente se haya involucrado con el proyecto en un mayor grado
- Entrega de un producto funcional al finalizar cada sprint
- Visualización del proyecto día a día
- Permite superar satisfactoriamente los fallos presentados durante el tiempo de vida del proyecto

B. DESVENTAJAS DE SCRUM

Cómo cualquier metodología de desarrollo, presenta algunas restricciones durante su aplicación:

- No genera toda la evidencia o documentación de otras metodologías
- No es apto para todos los proyectos, sobre todo para aquellos en los que intervienen equipos dispersos
- Es probable que sea necesario complementarlo con otras metodologías como XP

C. DIFERENCIAS DEL DESARROLLO TRADICIONAL VS. SCRUM

En el gráfico N° 2.2 se aprecia los principales contrastes que diferencian el desarrollo tradicional del desarrollo ágil Scrum.

Gráfico N° 2. 2
Desarrollo tradicional vs. Desarrollo ágil



Fuente: http://www.navegapolis.net/files/s/NST-010_01.pdf

Elaboración: Desarrolladores ágiles. Universidad Autónoma de México

Del gráfico de la página anterior, en cuanto al desarrollo ágil no lo realizan equipos diferentes especializados. Es un equipo único, formado por personas muy competentes, con perfiles y conocimientos que cubren las disciplinas necesarias para llevar a cabo el trabajo.

Respecto a las fases, se destaca que en el modelo Scrum no hay fases. En realidad las fases pasan a ser tareas que se ejecutan cuando se necesitan. No se hace primero el diseño del concepto o los requisitos, más tarde el análisis, luego el desarrollo, etc. Lo que aplicado al software serían las fases de: Requisitos del sistema, requisitos del software, análisis, diseño, construcción, pruebas e integración; y se ejecutarían de forma secuencial, pasan a tareas que se llevan a cabo en el momento que hacen falta. Normalmente a lo largo de pequeñas iteraciones durante todo el desarrollo.

Acerca de los requisitos iniciales, no se espera a disponer de requisitos detallados para comenzar el análisis, ni a tener éste para pasar a la codificación. Muchas veces los requisitos no se pueden conocer si no avanza el desarrollo y se va viendo y “tocando” el resultado. Otras veces el mercado es tan rápido que a mitad de trabajo las tendencias o la competencia obligarán a modificar el producto. Además, la participación de todo el equipo en el diseño aporta gran cantidad de talento innovador; un valor clave en el mercado de productos y servicios TIC.

Asimismo, los equipos ágiles empiezan a trabajar sin conocer con detalle cómo será el producto final. Parten de la visión general, y sobre ella, producen regularmente incrementos de funcionalidad que incrementan el valor al producto. Es decir, se van adaptando a los cambios.

2.2.6. OBJETIVOS Y PREFERENCIAS DE LA GESTIÓN ÁGIL

La gestión ágil de proyectos tiene como objetivo dar garantías a las demandas principales de la industria actual, entre los que tenemos:

A. VALOR

La gestión ágil se necesita en los mercados rápidos. Su objetivo es dar el mayor valor posible al producto, cuando éste se basa en: Innovación y flexibilidad.

La innovación, la permanencia de estas empresas depende de su capacidad de innovación continua. Del lanzamiento continuo de novedades, que compiten con los productos de otras empresas que también están en continua innovación.

La flexibilidad, el producto no sólo es valioso por su valor en el momento de su lanzamiento, sino también por su capacidad de adaptación y evolución a través de actualizaciones y ampliaciones.

B. REDUCCIÓN DEL TIEMPO DE SALIDA AL MERCADO

En la década de los 90, el tiempo medio de salida al mercado de los nuevos productos en EE.UU. se redujo de 35,5 a 11 meses (Wujec & Muscat, 2002). Este tiempo es un factor competitivo clave en determinados sectores.

Las estrategias de la gestión ágil para producir resultados en menos tiempo que la gestión tradicional son:

- Solapamiento de las fases de desarrollo
- Entrega temprana de las primeras partes del producto, que corresponden con las de mayor urgencia para el cliente, de forma que puede lanzar la primera versión en el menor tiempo posible.

C. AGILIDAD

Capacidad para producir partes completas del producto en periodos breves de tiempo.

D. FLEXIBILIDAD

Capacidad para adaptar la forma y el curso del desarrollo a las características del proyecto, y a la evolución de los requisitos.

E. RESULTADOS FIABLES

El objetivo de la gestión predictiva es ejecutar el trabajo planificado (y conocido de antemano) en el plazo planificado y por el coste previsto.

La gestión ágil no tiene un carácter predictivo o de anticipación. No conoce de antemano el detalle del producto que va a desarrollar, y por eso su objetivo no es fiabilidad en el cumplimiento de los planes, sino en el valor del resultado.

Los procesos de la gestión tradicional son buenos cuando consiguen desarrollar de forma repetible los productos especificados en el tiempo y con los costes previstos.

Los procesos de la gestión ágil son buenos, cuando consiguen entregar de forma temprana y continua un valor innovador.

Las preferencias de la gestión ágil, a diferencia de la tradicional, se muestran en el manifiesto ágil:

1. La capacidad de respuesta al cambio, sobre el seguimiento de un plan.

2. Los productos que funcionan frente a especificaciones y documentaciones innecesarias
3. La colaboración con el cliente frente a la negociación contractual
4. A las personas y su interacción por encima de los procesos y las herramientas

2.2.7. CONTROL DE LA EVOLUCIÓN DEL PROYECTO

Scrum controla de forma empírica y adaptable la evolución del proyecto, empleando las siguientes prácticas de la gestión ágil:

- **Revisión de las Iteraciones**, al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se tarda en reconducir una desviación en el proyecto o en las circunstancias del producto
- **Desarrollo incremental**, durante el proyecto, las personas implicadas no trabajan con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.
- **Desarrollo evolutivo**, los modelos de gestión ágil se emplean para trabajar en entornos de incertidumbre e inestabilidad de requisitos. Intentar predecir en las fases iniciales cómo será el producto final, y sobre dicha predicción desarrollar el diseño y la arquitectura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces. Para qué predecir los estados finales de la arquitectura o del diseño si van a estar cambiando. En Scrum se toma a la inestabilidad como una premisa, y se adoptan técnicas de trabajo para permitir esa evolución sin degradar la calidad de la arquitectura que se irá generando durante el desarrollo.

El desarrollo Scrum va generando el diseño y la arquitectura final de forma evolutiva durante todo el proyecto. No los considera como productos que deban realizarse en la primera “fase” del proyecto.

- **Auto-organización**, durante el desarrollo de un proyecto son muchos los factores impredecibles que surgen en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos.

En Scrum los equipos son auto-organizados (no auto-dirigidos), con margen de decisión suficiente para tomar las decisiones que consideren oportunas.

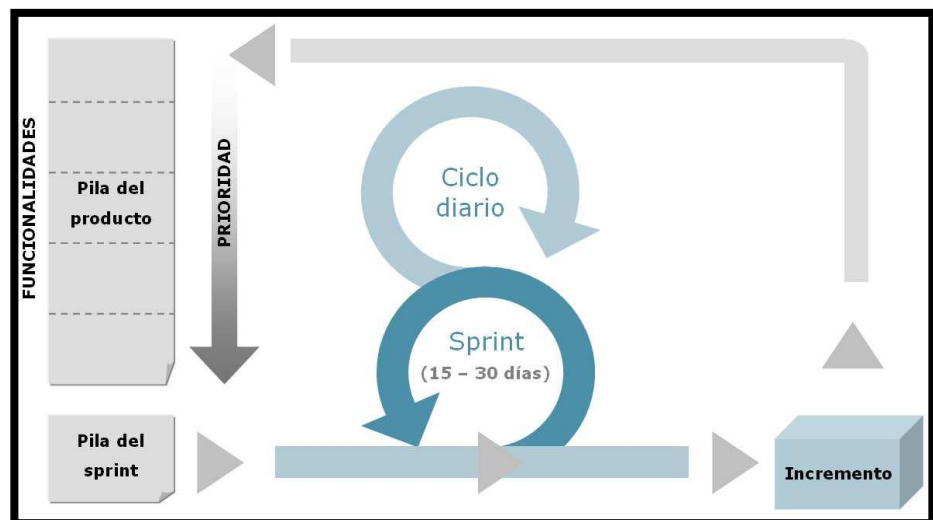
- **Colaboración**, las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo. Ésta es necesaria, porque para que funcione la auto organización como un control eficaz cada miembro del equipo debe colaborar de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

2.2.8. VISIÓN GENERAL DEL PROCESO

Scrum denomina “sprint” a cada iteración de desarrollo y recomienda realizarlas con duraciones de 30 días.

En el gráfico N° 2.3 se observa una visión general del proceso de desarrollo de software, siguiendo la Metodología Scrum.

Gráfico N° 2. 3
Visión general del proceso



Fuente: http://www.navegapolis.net/files/s/NST-010_01.pdf

Elaboración: Desarrolladores ágiles. Universidad Autónoma de México

Del gráfico, es importante considerar que el sprint es por tanto el núcleo central que proporciona la base de desarrollo iterativo e incremental.

Los elementos que conforman el desarrollo Scrum son:

A. LAS REUNIONES

- **Planificación de sprint:** Jornada de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en esa iteración.
- **Reunión diaria:** Breve revisión del equipo del trabajo realizado hasta la fecha y la previsión para el día siguiente.
- **Revisión de sprint:** Análisis y revisión del incremento generado.

B. LOS ARTEFACTOS

Scrum define una pequeña cantidad de artefactos para el seguimiento del proyecto y control de las actividades asociadas al sprint

- Product backlog (Pila del producto)
- Sprint backlog (Pila del sprint)
- Gráfica de progreso

C. LOS ROLES

Scrum clasifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto en: Dueño del Producto, Equipo Scrum, Scrum Manager o Scrum Master y “otros interesados”.

- **Equipo Scrum (Scrum Team):** Definido dentro del PMBOK como miembros del equipo de proyecto; se halla conformado por las personas dedicadas al desarrollo y diseño de la aplicación. Normalmente está constituido por un grupo de 5 a 9 individuos.
- **Dueño del Producto (Product Owner):** Cumple los roles de cliente y de usuario definidos dentro del PMBOK. La persona que asume este rol puede ser tanto un cliente como un miembro del equipo especializado en la lógica del negocio y procesos que se desarrollan; se encarga de proporcionar al equipo Scrum con conocimientos relacionados al negocio. Asimismo, administra los backlogs de los productos, que son un listado de requerimientos pendientes en donde todas las especificaciones del producto están enumeradas, los cuales están a la vista de todos los miembros de la organización.
- **Scrum Master:** Definido dentro del PMBOK como Director del Proyecto; esta persona tiene diariamente breves reuniones, denominados Scrums Diarios, con el equipo. Asimismo es el encargado de canalizar la información referente a nuevos requerimientos o modificaciones con la finalidad de disminuir el número de interrupciones a los desarrolladores.⁴

⁴ Cohn, M. D, (2008). Análisis, Diseño e Implementación de una Aplicación para la Administración de las Herramientas de Seguridad en una Red Local. *Tesis para optar el Título de Ingeniero Informático, Facultad de Ciencias e Ingeniería*. Perú.

Al final de cada ciclo (sprint) y previamente a comenzar el siguiente, el Scrum Master lleva a cabo una reunión (Scrum Retrospective), con los integrantes del equipo en donde se evalúan las experiencias vividas a lo largo del sprint y las conclusiones a las que se ha llegado.

Los tres primeros grupos (Dueño del Producto, Equipo Scrum y Scrum Master) son los responsables del proyecto, los que según la comparación siguiente (y sin connotaciones peyorativas) serían los “cerdos”; mientras que el resto de interesados serían las gallinas.

Cerdos y gallinas.

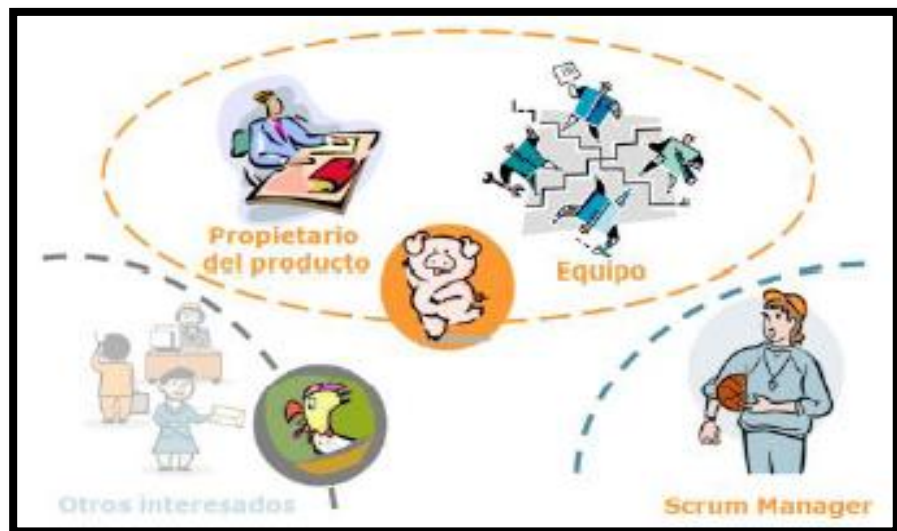
Esta metáfora ilustra de forma muy gráfica la diferencia entre implicancia y compromiso en el proyecto entre ambos grupos:

“Una gallina y un cerdo paseaban por la carretera. La gallina dijo al cerdo: “Quieres abrir un restaurante conmigo”.

El cerdo consideró la propuesta y respondió: “Sí, me gustaría. ¿Y cómo lo llamaríamos?”. La gallina respondió: “Huevos con beicon”. El cerdo se detuvo, hizo una pausa y contestó: “Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada”.

El gráfico N° 2.4 muestra la distribución clásica de los roles para Scrum.

Gráfico N° 2. 4 Distribución clásica de los roles para Scrum



Fuente: http://www.navegapolis.net/files/s/NST-010_01.pdf

Elaboración: Desarrolladores ágiles. Universidad Autónoma de México

En el gráfico se observan los principales roles de los involucrados en el desarrollo de un proceso aplicando la Scrum.

D. Valores

Scrum es una “carrocería” para dar forma a los principios ágiles. Es una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil: Cristal, DSDM, etc.

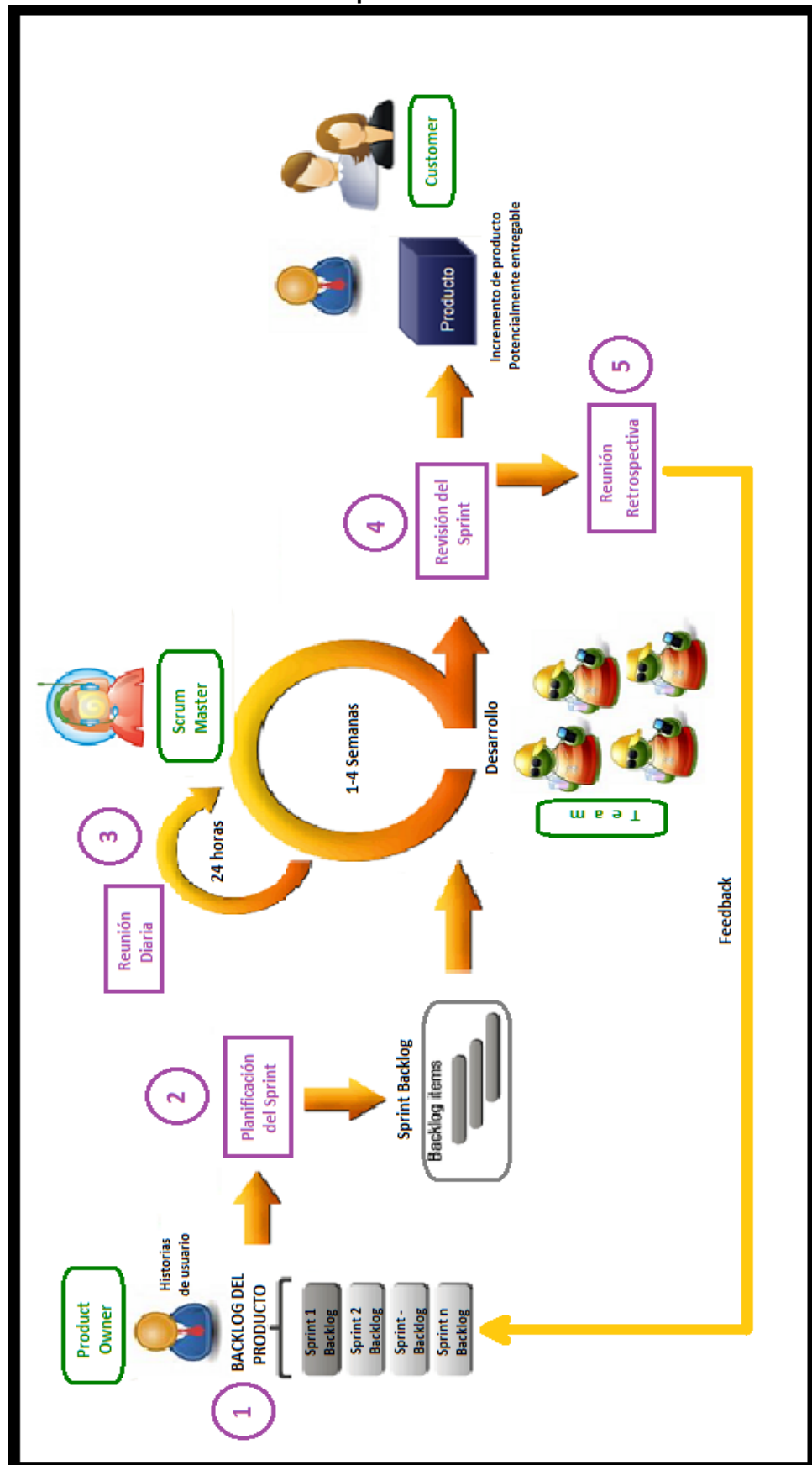
La carrocería sin motor, sin los valores que dan sentido al desarrollo ágil, no funciona.

- Delegación de atribuciones (empowerment) al equipo para que pueda auto-organizarse y tomar las decisiones sobre el desarrollo.
- Respeto entre las personas. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.
- Responsabilidad y auto-disciplina (no disciplina impuesta).
- Trabajo centrado en el desarrollo de lo comprometido
- Información, transparencia y visibilidad del desarrollo del proyecto.

2.3. MODELO APLICATIVO

El gráfico N° 2.5 permite visualizar el modelo aplicativo Scrum, que presenta la secuencia de pasos metodológicos que se deben seguir para poder aplicar la Metodología Scrum.

Gráfico N° 2. 5
Modelo aplicativo Scrum



Basado en: Rodríguez González, Pilar (2008). Estudio de la Aplicación de Metodologías Ágiles para la Evolución de productos software. Madrid, España. Pág. 17.

Del gráfico de la página anterior, se observa que el Modelo Scrum está conformado por las siguientes fases (para mayor detalle referirse al Anexo I):

- ✓ **FASE N° 1: DEFINIR BACKLOG DEL PRODUCTO**
- ✓ **FASE N° 2: PLANIFICACIÓN DEL BACKLOG**
- ✓ **FASE N° 3: SCRUM DIARIO**
- ✓ **FASE N° 4: REVISIÓN DEL SPRINT**
- ✓ **FASE N° 5: RETROSPECTIVA DEL SPRINT**

Independientemente del tipo de metodología que se utilice, cualquier desarrollo de software parte siempre de un mismo problema: conocer las necesidades de los clientes. Scrum, al igual que el resto de metodologías ágiles, pretende no centrar las tareas de desarrollo en un conjunto de requisitos formalmente definidos sino que aboga por la incorporación del cliente como un miembro más del equipo de desarrollo. De este modo, no se considera el proceso de definición de requisitos como un fin dentro del desarrollo del proyecto, sino que los requisitos aparecen implícitamente dentro del contenido de las denominadas historias de usuario.

A continuación se procede a detallar cada fase del modelo aplicativo Scrum:

FASE N° 1: DEFINICIÓN DEL BACKLOG DEL PRODUCTO

En esta primera fase, punto 1 del gráfico N° 2.5, antes de comenzar el primer sprint, es necesaria la elaboración del Backlog del Producto o Pila del Producto.

La pila de producto es el corazón de Scrum, es donde empieza todo. Básicamente es una lista priorizada de requisitos, historias o funcionalidades que el cliente desea, descritas en terminología del cliente, Se llama a esto historias de usuario, o a veces simplemente elementos de la pila, que por lo general, incluyen los siguientes campos:

- ID – un identificador único, simplemente un número auto-incremental. Esto permite no perder la pista a las historias cuando se cambia su nombre.
- Nombre – una descripción corta de la historia. Por ejemplo, “Ver tu historial de transacciones”. Suficientemente claro como para que el Dueño de Producto comprenda aproximadamente de qué se está hablando, y suficientemente clara como para que se distinga de las otras historias. Normalmente, 2 a 10 palabras.
- Importancia – el ratio de importancia que el Dueño de Producto da a ésta historia. Por ejemplo, 10. O 150. Más alto = más importante. Es preferible evitar el término “prioridad” porque típicamente “1” se considera la “máxima prioridad, lo que es muy incómodo si posteriormente se decide que algo es

más importante. ¿Qué prioridad se le daría a ese nuevo elemento? ¿Prioridad 0? ¿Prioridad -1?

- Estimación inicial – la valoración inicial del Equipo acerca de cuanto trabajo es necesario para implementar la historia, comparada con otras historias. La unidad son “puntos de historia” y usualmente corresponde a “días-persona ideales”. Se pregunta al equipo: “si tuvieras el número óptimo de personas para esta historia (ni muchos ni pocos, típicamente 2) y los encerraras en una habitación con cantidad de comida, y trabajan sin distracciones, ¿en cuántos días saldrían con una implementación terminada, demostrable, testeada y liberable?”. Si la respuesta es “con 3 personas encerrados en una habitación nos llevaría 4 días”, entonces la estimación inicial son 12 puntos (3x4). Lo importante no es que las estimaciones absolutas sean correctas (es decir, que una historia de 2 puntos deba durar 2 días), lo importante es que las estimaciones relativas sean correctas (es decir, que una historia de 2 puntos debería durar la mitad que una historia de 4 puntos).
- Como probarlo – una descripción a alto nivel de cómo se demostrará esta historia en la demo al final del Sprint. Se trata, esencialmente, de una simple especificación de un test: “Haz esto, entonces haz lo otro, y entonces debería ocurrir aquello”.
- Notas – cualquier otra información, clarificación, referencia a otras fuentes de información, etc. Normalmente muy breve.

El gráfico N° 2.6 muestra un ejemplo de pila de producto del lado del dueño del producto.

Gráfico N° 2. 6
Ejemplo de Product Backlog o Pila de Producto

Pila de Producto (ejemplo)					
ID	Nombre	Imp.	Est.	Como probarlo	Notas
1	Depósito	30	5	Entrar, abrir página de depósito, depositar 10€, ir a página de balance y comprobar que se ha incrementado en 10€	Necesita un diagrama UML. No preocuparse por encriptación aun
2	Ver tu historial de transacciones	10	8	Entrar, ver transacciones. Realizar un depósito de 10€. Ir a transacciones y comprobar que se ha actualizado con el nuevo depósito	Utilizar paginación para no hacer consultas muy grandes a la BB.DD. Diseño similar a la página de usuario.

Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 18.

Estos seis campos son los únicos que se utilizan sprint tras sprint. Se mantiene esta tabla en un documento Excel con la propiedad compartir habilitado (es decir, muchos usuarios pueden editar simultáneamente la hoja). Oficialmente, el dueño de producto es el propietario del documento, pero no se quiere dejar al resto de usuarios fuera. Muchas veces un desarrollador necesita abrir el documento para clarificar algo o cambiar una estimación. Por la misma razón, no se coloca este documento en el repositorio de control de versiones; en vez de eso, se almacena en una unidad de red compartida.

CAMPOS DE HISTORIAS ADICIONALES

A veces se usan campos adicionales en la pila de producto, fundamentalmente como comodidad para el dueño de producto a la hora de decidir sus prioridades.

- Categoría – una categorización básica de la historia, por ejemplo “backoffice” o “optimización”. Así, el dueño de producto puede filtrar fácilmente “optimización” y cambiar todas las prioridades de este tipo a “baja”, etc.
- Componentes - usualmente implementado en la forma de “checkboxes” en el documento Excel, por ejemplo “base de datos, servidor, cliente”. Aquí, el dueño de producto puede identificar qué componentes técnicos estarán involucrados en la implementación de la historia. Esto es útil cuando tienes varios equipos Scrum, por ejemplo un equipo de backoffice y otro equipo de cliente, y quieres que sea fácil para cada equipo saber a qué historias deben dedicarse.
- Solicitante – el dueño de producto puede querer mantener un historial acerca de qué cliente o persona interesada pidió originalmente la historia, para poder así ofrecerle información actualizada sobre el progreso de la misma.
- Bug tracking ID – si tienes un sistema de bug tracking (seguimiento de errores) aparte, es útil mantener un historial de cualquier correspondencia directa entre una historia y uno o más errores reportados.

COMO SE MANTIENE LA PILA DE PRODUCTO A NIVEL DE NEGOCIO

Si el dueño de producto tiene una formación técnica, puede que añada historias del tipo “añadir índices a la tabla de eventos”. ¿Por qué quiere algo así? El auténtico objetivo subyacente probablemente será algo como aligerar el formulario de búsqueda de eventos en el backoffice.

Puede ocurrir que los índices no fueran el cuello de botella que hiciera al formulario ir lento. Puede que fuera algo completamente diferente. El equipo está normalmente

mejor capacitado para averiguar cómo resolver algo, así que el Dueño de Producto debería concentrarse en los objetivos de negocio.

Cuando se encuentran historias con orientación técnica, es recomendable hacer al Dueño de Producto una serie de preguntas tipo “pero ¿Por qué?” hasta que encuentre el objetivo subyacente. Entonces, se reformula la historia en términos del objetivo subyacente (“aligerar el formulario de búsqueda de eventos del backoffice”). La descripción técnica original acaba siendo una nota (“indexar la tabla de eventos podría resolver esto”).

FASE N° 2: PLANIFICACIÓN DEL SPRINT

Esta segunda fase, es el punto 2 del gráfico N° 2.5. Cuando el día de la Planificación de Sprint se aproxima, una lección que se ha aprendido una y otra vez es: Asegurarse que la pila de producto está perfectamente lista antes de la reunión de planificación de Sprint. ¿Y esto qué significa? ¿Qué todas las historias deban estar perfectamente bien definidas? ¿Qué las estimaciones sean correctas? ¿Qué todas las prioridades hayan sido fijadas? No, lo que significa realmente es:

- La pila de producto debe existir
- Debería haber una Pila de Producto y un dueño de producto (por producto).
- Todos los elementos importantes deberían tener ratios de importancia asignados, diferentes ratios de importancia. En realidad, da igual si los elementos menos importantes tienen el mismo valor, ya que probablemente no se discutirán durante la planificación de Sprint en cualquier caso. Cualquier historia sobre la que el dueño de producto piense que tiene una remota posibilidad de incluirse en el Sprint debería tener un nivel de importancia único definido. El ratio de importancia se emplea sólo para ordenar los elementos por relevancia. Así que si el elemento A tiene una importancia de 20 y el elemento B una importancia de 100, simplemente significa que B es más importante que A. No significa que B sea cinco veces más importante que A. Si B tuviera una importancia de 21, ¡aun significaría lo mismo! Es útil dejar espacio entre la secuencia de números por si aparece un elemento C que es más importante que A pero menos importante que B. Por supuesto, le podríamos dar un ratio de importancia de 20,5 a C, pero queda mal, así que en vez de ello dejamos espacio entre números.
- El dueño de producto debe comprender cada historia (normalmente él es el autor, pero en algunos casos otras personas añaden solicitudes, que el dueño de producto puede priorizar). No necesita saber cómo exactamente debe implementarse, pero debería entender por qué la historia está ahí.

Nota: Otras personas aparte del dueño de producto pueden añadir sus historias a la pila de producto. Pero no pueden asignarles niveles de importancia, ese es un cometido exclusivo del dueño de producto. Tampoco pueden establecer estimaciones, ese es un cometido exclusivo del equipo.

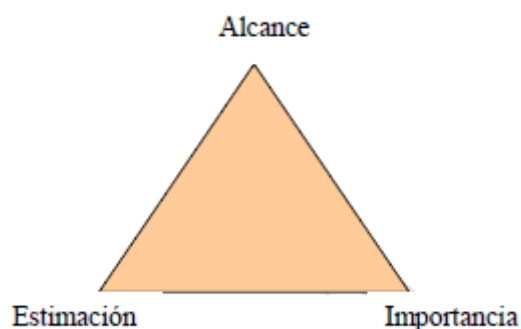
La planificación de sprint es una reunión crítica, probablemente la más importante de Scrum. Una planificación de Sprint mal ejecutada puede arruinar por completo todo el Sprint. El propósito de la planificación de Sprint es proporcionar al equipo suficiente información como para que puedan trabajar en paz y sin interrupciones durante unas pocas semanas, y para ofrecer al dueño de producto suficiente confianza como para permitírselo.

Una planificación de Sprint produce, concretamente:

- Una meta de Sprint
- Una lista de miembros (y su nivel de dedicación, si no es del 100%)
- Una Pila de Sprint o Sprint Backlog (lista de historias incluidas en el Sprint)
- Una fecha concreta para la Demo del Sprint
- Un lugar y momento definidos para el Scrum Diario

Es importante que el dueño de producto asista a la planificación del sprint, se sabe que a veces los dueños de producto se resisten a pasar horas con el equipo preparando la planificación de Sprint. “Miren, ya les he listado lo que quiero. No tengo tiempo para estar en la reunión de planificación”. Es un problema bastante serio. La razón por la que el equipo y el dueño de producto deben asistir a la planificación de sprint es que cada historia contiene tres variables que son muy dependientes unas de otras, tal como se muestra en el gráfico N° 2.7.

Gráfico N° 2. 7
Variables involucradas en la definición de una historia de usuario



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 23.

Del gráfico de la página anterior, el alcance y la importancia los fija el dueño de producto. La estimación la proporciona el equipo. Durante una planificación de Sprint, estas variables sufren un ajuste fino y continuo a través del diálogo cara a cara entre el equipo y el dueño de producto.

La razón por la que todo el equipo y el dueño de producto tienen que estar en la planificación es porque cada historia contiene tres variables que son altamente dependientes la una de la otra.

Normalmente, el dueño de producto comienza la reunión resumiendo cuál es su meta para el sprint y las historias más importantes. A continuación, el equipo las repasa y les asigna una estimación, comenzando con la más importante.

Conforme se va haciendo, aparecerán dudas importantes respecto al alcance: “esta historia sobre ‘borrar usuario’, ¿incluye repasar todas las transacciones pendientes el usuario y cancelarlas?”. En algunos casos, las respuestas sorprenderán al equipo y les obligarán a cambiar sus estimaciones.

En algunos casos, la estimación para una historia no será la que el dueño de producto esperaba. Esto puede forzarle a cambiar la importancia de la historia o su alcance, o que obligará al equipo a re-estimarla, etc., etc. Este tipo de colaboración directa es fundamental en Scrum.

En el triángulo mencionado anteriormente, también se debe considerar una cuarta variable: la calidad. Se puede distinguir entre calidad interna y calidad externa.

- Calidad externa: es lo que perciben los usuarios del sistema. Un interfaz de usuario lento y poco intuitivo es un ejemplo de baja calidad externa.
- Calidad interna: se refiere a aquellos aspectos que normalmente no son visibles al usuario, pero que tienen un profundo efecto en la mantenibilidad del sistema. Cosas como consistencia del diseño del sistema, cobertura de pruebas, legibilidad del código, refactorización, etc.

AGENDA DE LA REUNIÓN DE PLANIFICACIÓN DE SPRINT

Tener algún tipo de agenda u orden del día de la reunión de planificación de Sprint reducirá el riesgo de sobrepasar la duración determinada.

El gráfico N° 2.8 muestra un ejemplo típico de una agenda.

Del gráfico de la página siguiente se puede acotar que la agenda no es en absoluto inamovible. El Scrum Master puede ampliar o acortar los periodos según sea necesario conforme progresa la reunión.

Gráfico N° 2. 8
Ejemplo típico de una agenda de reunión Scrum

Reunión de planificación de Sprint: 13:00 – 17:00 (10 minutos de descanso cada hora)

- **13:00 – 13:30.** El Dueño de Producto comenta la meta del Sprint y resume la Pila de Producto. Se establece un lugar, fecha y hora para la Demo.
- **13:30 – 15:00.** El equipo da estimaciones de tiempo, y divide los elementos tanto como sea necesario. El Dueño de Producto actualiza los ratios de importancia. Se clarifican los elementos. Para todos los elementos de alta importancia se establece el “Como probarlo”.
- **15:00 – 16:00.** El equipo selecciona las historias que se incluirán en el Sprint. Se realizan cálculos de velocidad para chequear si es factible.
- **16:00 – 17:00.** Se selecciona un lugar y hora para el Scrum Diario (si es que cambio respecto al último Sprint). Se continúa dividiendo las historias en tareas.

Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 26.

DEFINIENDO LA DURACIÓN DEL SPRINT

Uno de los resultados de la planificación de sprint es una fecha de revisión (demo) de sprint definida. Eso significa que se debe determinar una duración del sprint. ¿Y cuánto debería durar un sprint? Bueno, los sprints cortos están bien. Permiten a la compañía ser “ágil”, es decir, cambiar de dirección frecuentemente. Sprints cortos = ciclo de feedback corto = más entregas y más frecuentes = más feedback del cliente = menos tiempo desarrollando en dirección incorrecta = aprender y mejorar más rápido, etc.

Pero los sprints largos tampoco están mal. El equipo tiene más tiempo para conseguir impulso, tienen más espacio para recuperarse de los problemas que surjan y aun así cumplir la meta del sprint, tiene menos carga de gestión en términos de reuniones de planificación de sprints, demos, etc.

Generalmente, los dueños de producto prefieren los Sprints cortos y a los desarrolladores les gustan los sprints largos. Así que la duración del sprint es un valor de compromiso. En base a la experiencia con varias duraciones, al final se ha encontrado la duración favorita: 3 semanas. La mayoría de los equipos (pero no todos) hacen sprints de 3 semanas (15 días). Suficientemente cortos para proporcionar agilidad corporativa, suficientemente largos para lograr *flujo* y recuperarse de los problemas que aparezcan durante el sprint.

DEFINIENDO LA META DEL SPRINT

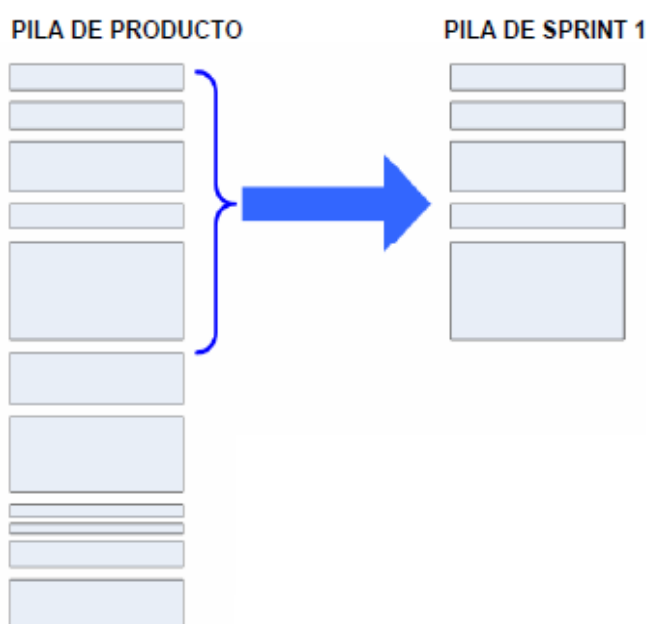
En algún momento durante la reunión de planificación de sprint, se pregunta “¿Y cuál es la meta del sprint?”. La meta de Sprint debería responder a la pregunta

fundamental “¿Por qué se hace este sprint en vez de irse todos de vacaciones?”. De hecho, una forma de obtener la meta del dueño de producto es precisamente hacerle esa misma pregunta.

Una de las principales actividades durante la planificación de Sprint es decidir qué historias se incluyen en el sprint. Más específicamente, qué historias de la pila de producto copiar en la pila de sprint.

En el gráfico N° 2.9, se muestra un esquema de cómo se compone una pila de producto, la cual a su vez se desglosa en una serie de sprints backlog.

Gráfico N° 2. 9
Esquema de Pila de producto



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 28.

Del gráfico de la página siguiente, se observa que cada rectángulo representa una historia, ordenadas por importancia. La historia más importante está al principio de la lista. El tamaño de cada rectángulo representa el tamaño de la historia (es decir, el tiempo estimado en puntos de historia). La altura del corchete azul representa la velocidad estimada del equipo, es decir, cuántos puntos de historia cree el equipo que pueda completar durante el próximo sprint.

La pila de sprint de la derecha es una instantánea de las historias de la pila de producto. Representa las historias a las que el equipo se compromete durante este Sprint. El equipo decide cuántas historias incluirá en el sprint. No el dueño de producto ni nadie más. Para decidir que historias incluir en el sprint, el equipo utiliza dos técnicas para esto:

1. A ojo de buen cubero: El ojo de buen cubero funciona bastante bien para equipos pequeños y sprints cortos.
2. Cálculos de velocidad: Esta técnica consta de dos pasos: primero, decidir la velocidad estimada. Y segundo, calcular cuántas historias se pueden añadir sin sobrepasar la velocidad estimada.

La velocidad es una medida de “cantidad de trabajo realizado”, donde cada elemento se evalúa en función de su estimación inicial.

El gráfico N° 2.10 muestra un ejemplo de velocidad estimada al principio de un sprint y la velocidad real al final de dicho sprint. Cada rectángulo es una historia, y el número interior es la estimación inicial de dicha historia.



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 32.

Del gráfico se observa que la velocidad real está basada en las estimaciones iniciales de cada historia. Cualquier actualización a la estimación de la historia realizada durante el sprint es ignorada.

Una manera muy fácil de estimar la velocidad es revisar la historia del equipo. ¿Cuál fue su velocidad durante los últimos sprints? Y entonces asumir que la velocidad será más o menos la misma en el próximo Sprint. Esta técnica se conoce como “el tiempo que hizo ayer”. Solo es factible para equipos que ya han hecho algunos sprints (de forma que haya estadísticas disponibles) y que harán el próximo sprint más o menos de la misma manera, con el mismo tamaño de equipo, las mismas condiciones de trabajo, etc. Este, claro está, no siempre es el caso. Una forma más sofisticada de hacerlo es realizar un simple cálculo de recursos. Un ejemplo sería el siguiente: “Digamos que se está planificando un Sprint de 3 semanas (15 días laborables) con

un equipo de 4 personas. Lisa estará de vacaciones 2 días. Dave sólo estará disponible al 50% y estará un día de vacaciones”. Todo ello resulta:

Días disponibles

Tom	15
Lisa	13
Sam	15
Dave	7

50 DÍAS-HOMBRE disponibles

Se tiene 50 días-hombre disponibles en este sprint. ¿Es esta la velocidad estimada? No. Porque la unidad de estimación son puntos de historia lo que, en este caso, corresponde más o menos a “días-hombre ideales”. Un día-hombre ideal es un día perfectamente efectivo, sin distracciones, lo cuál es raro. Lo que es más, se deben tener en cuenta cosas como trabajo no planificado que se añade al sprint, gente que se pone enferma, etc.

Así que la velocidad estimada será sin duda menor de 50. ¿Pero cuánto menor? Para esto se usa el “factor de dedicación”.

VELOCIDAD ESTIMADA DE SPRINT

(DÍAS-HOMBRE DISPONIBLES) X (FACTOR DE DEDICACIÓN) = VELOCIDAD ESTIMADA
--

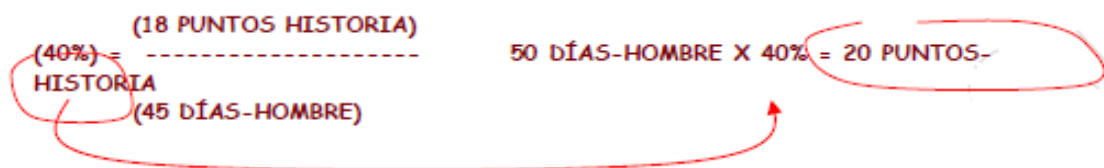
El factor de dedicación es una estimación de cuán centrado va a estar el equipo. Un factor de dedicación bajo puede significar que el equipo espera encontrar muchas distracciones e impedimentos o que considera que sus propias estimaciones son optimistas.

La mejor manera de determinar un factor de dedicación razonable es estudiar el último sprint (o incluso mejor, la media de los últimos sprints).

FACTOR DE DEDICACIÓN DEL ÚLTIMO SPRINT

$\text{FACTOR DE DEDICACIÓN} = \frac{(\text{VELOCIDAD REAL})}{(\text{DÍAS-HOMBRE DISPONIBLES})}$
--

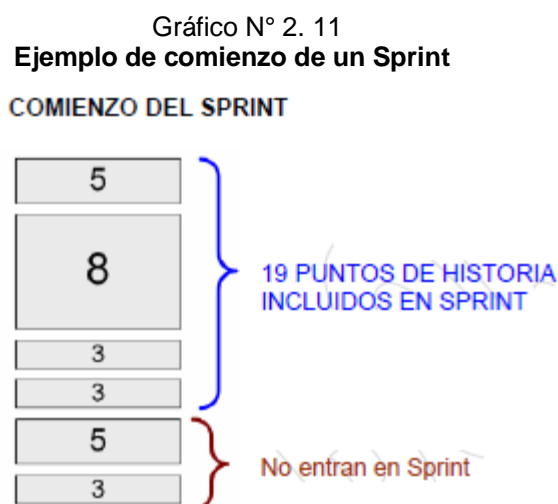
La velocidad real es la suma de las estimaciones iniciales que se completaron en el último sprint. Suponiendo que en el último sprint se completaron 18 puntos de historia utilizando un equipo de 3 personas formado por Tom, Lisa y Sam trabajando las tres semanas hasta un total de 45 días-hombre. Y ahora se intenta calcular la velocidad del próximo sprint. Para complicar las cosas, un nuevo miembro, Dave, se une al equipo para este sprint. Teniendo en cuenta las vacaciones y demás asuntos se tiene 50 días-hombre ideales este Sprint.



Así que la velocidad estimada para el próximo Sprint es de 20 puntos de historia. Eso significa que el equipo debe añadir historias al Sprint hasta que sume aproximadamente 20.

COMIENZO DEL SPRINT

En el gráfico N° 2.11, se muestra un esquema del comienzo del Sprint.



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 34.

En este caso, el equipo puede escoger las 4 historias más importantes hasta un total de 19 puntos de historia, o las 5 historias más importantes hasta 24 puntos de historia. Suponiendo que escogen 4 historias, ya que se aproximan más a los 20 puntos de historia. Siempre que haya dudas, se escoge añadir menos historias. Dado

que estas 4 historias suman 19 puntos, la velocidad estimada final para este Sprint es de 19.

El tiempo que hizo ayer es una técnica sencilla, pero se debe usar con cierta dosis de sentido común. Si el último sprint fue especialmente malo porque la mayoría del equipo estuvo enfermo una semana, entonces podría ser adecuado asumir que no se volverá a tener tan mala suerte y se podría estimar un factor de dedicación mayor el próximo Sprint. Si el equipo ha instalado recientemente un sistema súper rápido de compilación continua probablemente también podrás incrementar el factor de dedicación gracias a ello. Si una nueva persona se une a este sprint se debería reducir su factor de dedicación para tener en cuenta su formación, etc.

Siempre que sea posible, se debe tener en cuenta varios sprints y sacar medias para conseguir estimaciones más acertadas.

Tras la reunión de planificación de sprint, el Scrum Master debe actualizar manualmente la pila de producto en excel respecto a cualquier cambio que se haya realizado.

La estimación es una labor de equipo, todos los miembros del equipo deben involucrarse en estimar cada historia, ello debido a que:

- A la hora de planificar, normalmente no se sabe exactamente quién implementará qué partes de cada historia.
- Las historias normalmente involucran a bastantes personas y de diferentes áreas de experiencia (diseño de interfaz de usuario, programación, pruebas, etc.).
- Para poder proporcionar una estimación, un miembro del equipo necesita comprender de alguna forma de qué trata la historia. Pidiendo a todo el mundo que estime la historia se asegura que cada miembro del equipo comprende de qué trata cada elemento. Esto incrementa las posibilidades de que unos miembros del equipo ayuden a otros durante el sprint. También mejora las posibilidades de que aparezcan pronto las preguntas importantes sobre cada historia.

DIVIDIENDO LAS HISTORIAS EN TAREAS

Para diferenciar historias y tareas, es necesario comprender que las historias son entregables de los que el dueño de producto se preocupa. Las tareas son no-entregables, o aspectos de los que el dueño de producto no se preocupa.

El gráfico N° 2.12, presenta un ejemplo de división de una historia en historias más pequeñas.

Del gráfico de la página siguiente, la historia “Gestión de Usuarios” se divide en dos pequeñas tareas: “Añadir / Modificar Usuario” y “Buscar Usuario”.

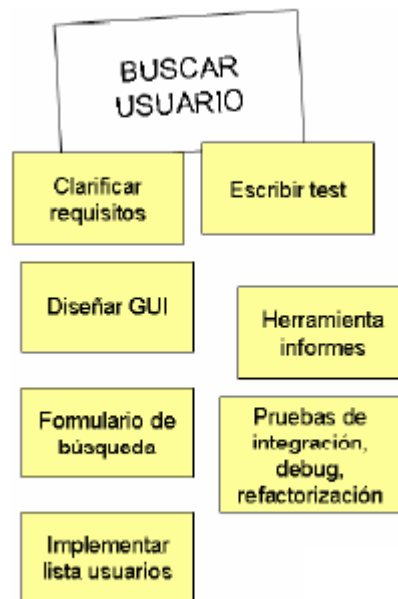
Gráfico N° 2. 12
Ejemplo de división de una historia en tareas



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 42.

El gráfico N° 2.13, presenta una subdivisión de la tarea “Buscar Usuario”.

Gráfico N° 2. 13
Ejemplo de subdivisión de una tarea



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 43.

Del gráfico, la tarea “Buscar Usuario” se subdivide en siete tareas, al completar las cuales se tendrá realizada la tarea principal.

FASE N° 3: SCRUM DIARIO

En esta tercera fase, punto 3 del gráfico N° 2.5, se deberá definir el sitio y la hora para el Scrum Diario.

Uno de los productos frecuentemente olvidados de la planificación de sprint es un sitio y una hora determinados para el Scrum Diario. Sin ello, tu sprint está condenado

a un mal comienzo. El primer scrum diario es esencialmente el lanzamiento, donde todo el mundo decide por dónde va a empezar a trabajar.

- Desventaja de scrums por las tardes, cuando se llega al trabajo por la mañana, se tiene que acordar de qué se le dijo a la gente sobre lo que deberían hacer hoy.
- Desventaja de los scrums por las mañanas, cuando se llegas al trabajo por la mañana, se debe acordar de qué se hizo ayer para informar sobre ello hoy.

El procedimiento por defecto es seleccionar la hora más temprana a la que ningún miembro del equipo vaya a quejarse. Usualmente las 9:00, 9:30 o 10:00a.m. Lo más importante es que sea a una hora a la que todo el equipo acepte con total convencimiento.

Si el tiempo se está agotando. De todos los asuntos que queremos resolver durante la planificación de sprint, ¿qué abandonar si se queda sin tiempo? Bueno, normalmente se usa la siguiente lista de prioridades:

Prioridad 1: Una meta de sprint y una fecha para la demo. Esto es lo mínimo que se necesita para comenzar un sprint. El equipo tiene una meta y una fecha de finalización, y puede trabajar directamente con la pila de producto. Se debe considerar seriamente organizar una nueva reunión de planificación de sprint mañana mismo, pero si realmente se necesita que el sprint comience entonces probablemente se pueda hacer con esto.

Prioridad 2: Lista de qué historias ha aceptado terminar el equipo en este sprint.

Prioridad 3: Una estimación para cada historia del sprint.

Prioridad 4: “Cómo probarlo”, relleno para cada historia del sprint.

Prioridad 5: Cálculos de velocidad y recursos, como chequeo de la planificación del sprint. Incluyendo una lista de los miembros del equipo y sus compromisos (de otra forma, no se podría calcular la velocidad).

Prioridad 6: Un sitio y hora específicos para la realización del scrum diario. Sólo se necesita un momento para decidirlo, pero si se queda sin tiempo el Scrum Master puede simplemente decidir esto después de la reunión y mandar un correo a todo el mundo.

Prioridad 7: Historias divididas en tareas. Esta división puede sin embargo hacerse diariamente durante los scrum diarios, pero interferirá levemente el flujo del sprint.

La reunión de planificación de Sprint tiene éxito si todo el mundo (todos los miembros del equipo y el dueño de producto) sale de la reunión con una sonrisa y se levanta a

la mañana siguiente con una sonrisa, y hacen su primer scrum diario con una sonrisa. A partir de ahí, por supuesto, toda clase de cosas pueden ir mal, pero al menos no puedes echarle la culpa al plan de sprint.

El gráfico N° 2.14, muestra la utilización de una página de información para comunicar los sprints.

Gráfico N° 2. 14
Ejemplo de página de información de Sprint

Equipo Jackass, Sprint 15

Objetivo de Sprint:

- ¡Versión beta lista!

Pila de Sprint:

- Deposito (3)
- Herramienta de migración (8)
- Login en backoffice (5)
- Administración de usuarios de backoffice (5)

Velocidad estimada: 21

Calendario:

- Periodo de Sprint: 6/11/2006 a 24/11/2006
- Scrum diario: 9:30-9:45 en la sala del equipo
- Demo de Sprint: 24/11/2006, 13:00 en la cafetería

Equipo:

- Jim
- Erica (Scrum Master)
- Tom (75%)
- Eva
- John

Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 48.

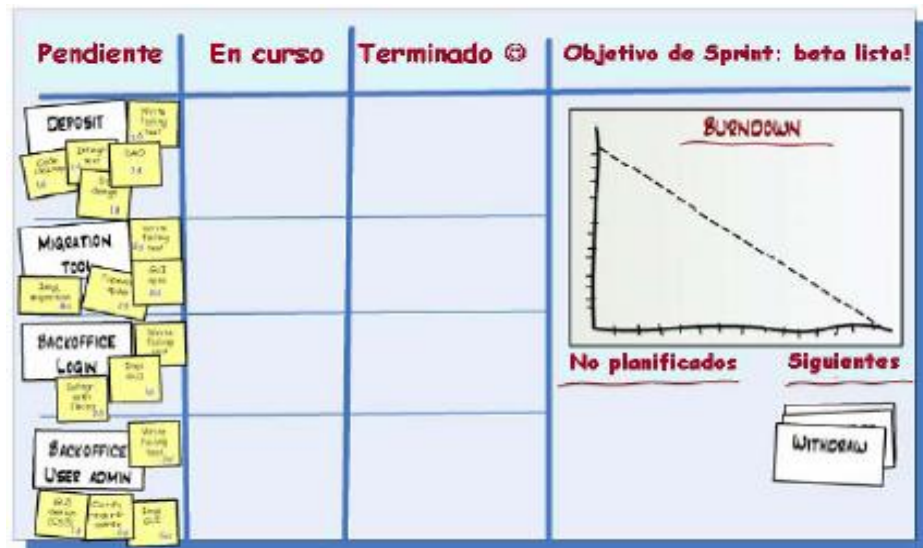
Del gráfico se deduce que es importante mantener a toda la compañía informada sobre lo que está ocurriendo. De otra forma, la gente se queja constantemente o, incluso peor, hacen falsas presunciones sobre lo que está ocurriendo.

FORMATO DE LA PILA DE SPRINT

El formato más efectivo para la pila de sprint, es una tabla de tareas en la pared. Para ello se debe buscar una pared que no esté usada o contenga cosas inútiles como el logo de la compañía, viejos diagramas o cuadros feos. Pegar una gran hoja de papel (por lo menos de 2x2 metros, o 3x2 para un equipo grande). Y entonces un cuadro como el que se visualiza en el gráfico N° 2.15.

Del gráfico de la página siguiente, se visualiza la estructura de una tabla de tareas que contiene los campos: Pendiente, en curso, terminado y objetivo de sprint.

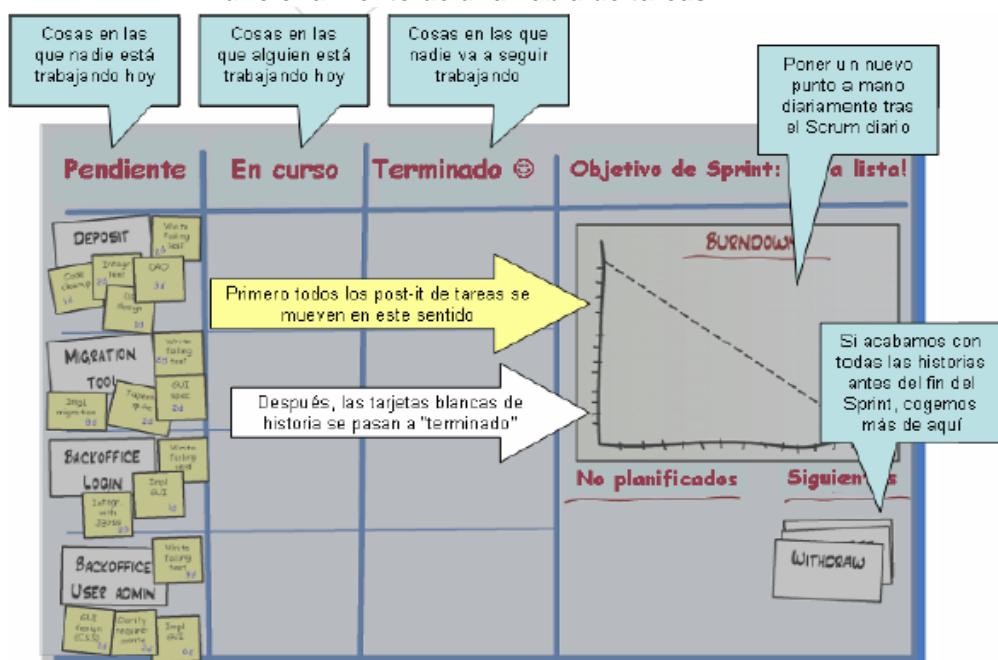
Gráfico N° 2. 15
Estructura de una Tabla de tareas



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 51.

Se podría usar una pizarra, pero es un desperdicio. Si es posible, se debe guardar las pizarras para garabatos de diseño y usar las paredes para los tabloneros de tareas. Si se usa post-it para las tareas, no olvidar pegarlos con cinta adhesiva o tomarle una fotografía, o se encontrará todos los pos-it en el suelo cualquier día. La tabla de tareas funciona tal y como se observa en el gráfico N° 2.16.

Gráfico N° 2. 16
Funcionamiento de una Tabla de tareas

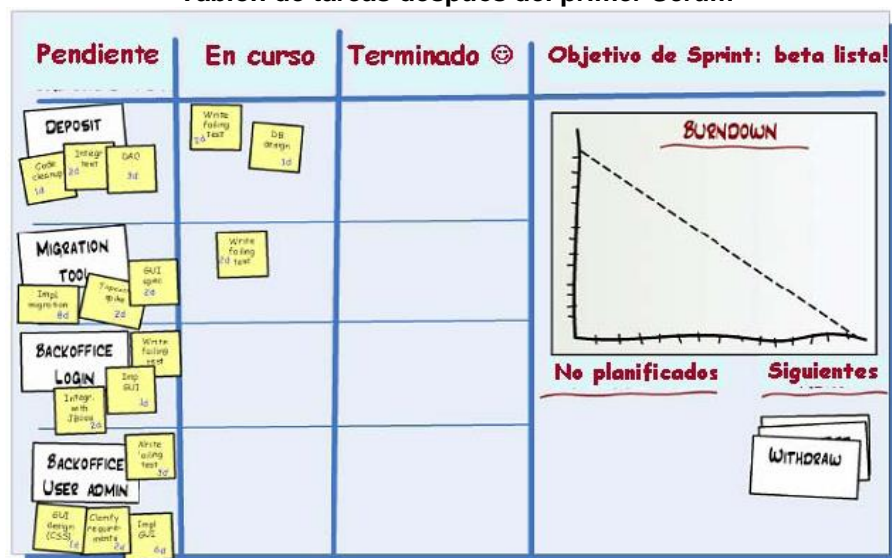


Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 51.

Del gráfico de la página anterior, es sobreentendido que se puede añadir toda clase de columnas adicionales. “Esperando test de integración”, por ejemplo. O “cancelado”. De todas formas, antes de complicar las cosas, se debe pensar profundamente. Tener en cuenta que la simplicidad es extremadamente valiosa en estas cosas, así que sólo se añade complicaciones adicionales si el coste de no hacerlo es demasiado grande.

Después del primer Scrum, el tablón puede quedar como en el gráfico N° 2.17.

Gráfico N° 2. 17
Tablón de tareas después del primer Scrum



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 52.

Del gráfico de la página anterior, se observa claramente que después de haber ocurrido el primer sprint, las tareas pasaron de la columna “Pendiente” a la columna “En curso”, para ser trabajadas por el equipo Scrum. Como se puede ver, tres tareas están en proceso, es decir, el equipo estará trabajando en estos elementos hoy.

A veces, para equipos más grandes, una tarea queda atascada “En progreso” porque nadie recuerda quién estaba trabajando en ella. Si esto ocurre a menudo en un equipo, usualmente introducimos políticas como etiquetar cada tarea en progreso con el nombre de la persona que la ha emprendido.

El gráfico N° 2.18 muestra la tabla de tareas unos días más tarde.

Del gráfico de la página siguiente, se ha completado la historia “Depósito” (es decir, ha sido chequeada en el repositorio de código fuente, testeada y refactorizada). La herramienta de migración (segunda historia) está parcialmente completada. La tercera historia (“backoffice login”) ha comenzado, y la cuarta (“backoffice user admin.”) no ha empezado aún.

Gráfico N° 2. 18
Tabla de tareas unos días más tarde



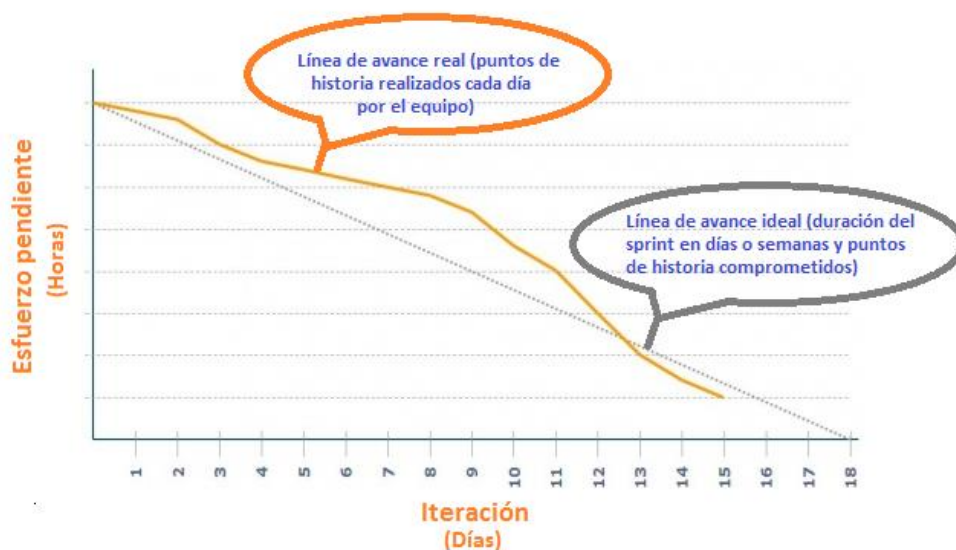
Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 53.

Se tienen tres elementos no planificados, como puede verse abajo a la derecha. Esto es útil para recordar cuando se haga la retrospectiva del Sprint.

COMO FUNCIONA EL DIAGRAMA BURNDOWN

El gráfico N° 2.19 muestra el diagrama burndown.

Gráfico N° 2. 19
Tablón de tareas después del primer Scrum



Tomado de: Kniberg, H. (2007). *Scrum and XP from the Trenches: How we do scrum*. Estados Unidos: Editorial C4Media. Page. 54.

El diagrama de la página anterior muestra que:

- En el primer día del Sprint, 1 de Agosto, el equipo estimó que habían aproximadamente 70 puntos de historia en los que trabajar. Esta era, consecuentemente, la *velocidad estimada* para todo el Sprint.
- El 16 de Agosto el equipo estima que quedan aproximadamente 15 puntos de historia por hacer. La línea de puntos muestra que se está incluso algo avanzado respecto a la planificación, es decir, que a este paso se completaría todo al final del Sprint. Se salta los fines de semana en el eje X ya que rara vez se trabaja los fines de semana. Además, se solía incluir los fines de semana, pero esto hace a los burndown algo confusos ya que se “aplanan” durante los fines de semana, lo que parecía una señal de peligro.

ESTIMACIÓN EN DÍAS VS. HORAS

En la mayoría de los libros y artículos sobre Scrum se encuentra que las tareas se estiman en horas, no en días. Se solía hacer eso, la fórmula general era:

$$1 \text{ día-hombre real} = 6 \text{ horas-hombre reales}$$

Ahora se ha dejado de hacer eso, al menos en la mayoría de los equipos, por las siguientes razones:

- Las estimaciones en horas-hombre eran demasiado granulares. Esto provocaba una tendencia a estimar muchas tareas en 1-2 horas, y con ello a la microgestión.
- De todas formas resultó que todo el mundo estaba pensando en términos de días-hombre, y simplemente multiplicaban por 6 para escribir las horas-hombre.
- Dos unidades diferentes causan confusión: “¿Esa es la estimación en días-hombre o en horas-hombre?”.

Así que ahora se usa días-hombre como base para todas las estimaciones (aunque se sigue llamando puntos de historia). El valor más bajo es 0.5, es decir, cualquier tarea que es menor de 0.5 se elimina o se combina con otras tareas, o se le deja una estimación de 0.5 (no causa mucho daño sobre-estimar un poco).

FASE N° 4: REVISIÓN DEL SPRINT

En esta cuarta fase, punto 4 del gráfico N° 2.5, se habla acerca de la Revisión del Sprint o también conocido como Demo de Sprint, que es una parte importante de Scrum que la gente tiende a subestimar.

La revisión del sprint, involucra la presencia del equipo Scrum, Scrum Master, Product Owner con todas las personas implicadas en el proyecto (gallinas). La duración máxima de ésta reunión es de cuatro horas, y el objetivo es presentar al propietario del producto y a las gallinas las nuevas funcionalidades implementadas.

Una demo de sprint bien ejecutada, aunque parezca poco espectacular, tiene un efecto muy profundo:

- El equipo obtiene reconocimiento por sus logros. Se sienten bien.
- Otras personas se enteran de lo que está haciendo el equipo.
- La demo consigue feedback vital de los interesados.
- Las demos son (o deberían ser) un evento social donde diferentes equipos pueden interactuar unos con otros y discutir su trabajo. Esto es muy valioso.
- Hacer una demo fuerza al equipo a acabar realmente las cosas y entregarlas (incluso aunque sea sólo en entorno de pruebas). Sin las demos, se sigue consiguiendo enormes montones de cosas terminadas al 99%. Con las demos puede que se consigan menos cosas terminadas, pero estas están realmente terminadas, lo que es mucho mejor que tener una enorme pila de cosas que están más o menos listas y que se pulirán en el próximo sprint.
- Si un equipo se ve obligado a hacer una demo de sprint, incluso aunque no tengan mucho que realmente esté funcionando, la demo será embarazosa. El equipo tartamudeará y tropezará mientras hace la demo y el aplauso de después será frío. La gente sentirá un poco de pena por el equipo, algunos se enfadarán con ellos por hacerles perder el tiempo con una demo pésima. Esto duele, pero el efecto es similar al de una amarga medicina. En el próximo sprint, el equipo intentará por todos los medios tener cosas terminadas. El equipo sabe que tendrán que hacer una demo pase lo que pase, lo que incrementa significativamente las posibilidades de que haya algo útil que demostrar.

LISTA DE COMPROBACIÓN PARA DEMOS DE SPRINT

- Se debe asegurar de presentar claramente el objetivo del Sprint. Si hay personas en la demo que no saben nada sobre tu producto, tómate un par de minutos para describirlo.
- No se debe perder mucho tiempo preparando la demo, especialmente en llamativas presentaciones. Se debe concentrar en mostrar código funcionando.

- Mantener el paso rápido, es decir, se debe concentrar la preparación en hacer que la demo sea rápida en lugar de bonita.
- Mantener la demo a nivel de negocio, dejar los detalles técnicos aparte. Se debe concentrar en “qué se ha hecho” en lugar de “cómo se ha hecho”.
- En la medida de lo posible, se debe dejar que la audiencia pruebe el producto por sí misma.
- No se debe mostrar un montón de pequeños errores solucionados y funcionalidades triviales. Se puede mencionar, pero no mostrar, ya que normalmente se tarda mucho y desvía la atención de las historias más importantes.

Al final de la reunión se interroga individualmente a todos los asistentes para recabar impresiones, sugerencias de cambio y mejora, y su relevancia. Y el propietario del producto trata con los asistentes y con el equipo las posibles modificaciones en el product backlog.

FASE N° 5: RETROSPECTIVA DEL SPRINT

En esta quinta fase, punto 5 del gráfico N° 2.5, lo más importante de una Retrospectiva de Sprint es asegurarse de que tienen lugar.

Por alguna razón, los equipos no siempre parecen inclinados a hacer retrospectivas. Sin un empujón, la mayoría de los equipos usualmente se saltarían la retrospectiva y continuarían con el próximo sprint. Pero, todo el mundo coincide en que las retrospectivas son extremadamente útiles. De hecho, se puede afirmar que la retrospectiva es el segundo evento más importante de Scrum (siendo el primero la reunión de planificación de sprint), ya que es la mejor oportunidad para mejorar.

Por supuesto, no se necesita una reunión de retrospectiva para conseguir buenas ideas, pero si la idea viene del equipo, es decir, surge durante la retrospectiva, entonces a todo el mundo se le permite contribuir y discutir las ideas. Sin las retrospectivas se notará que el equipo sigue cometiendo los mismos errores una y otra vez.

Para organizar las retrospectivas, el formato general varía un poco, pero normalmente se hace algo como esto:

- Se reserva 1-3 horas, dependiendo de cuánta discusión esperemos.
- Participantes: el Dueño de Producto, el Equipo Scrum y el Scrum Master.
- Se conducen a una reunión cerrada, un rincón cómodo con sofás, el patio del tejado o algún sitio similar. Que se pueda tener una discusión sin interrupciones.

utilizan “votación por puntos” para determinar en qué mejoras centrarse el próximo Sprint. Cada miembro del equipo tiene tres imanes y se les invita a votar sobre cualquier mejora en la que les gustaría trabajar en el próximo sprint. Cada miembro del equipo distribuye los imanes como quiera, incluso colocando los tres en el mismo elemento. Basándose en esto, se selecciona cinco mejoras de procesos en los que concentrarse, y se evalúa en la siguiente retrospectiva.

Es importante no ser demasiado ambicioso. Se debe concentrar en unas pocas mejoras en cada Sprint.

DIFUNDIENDO LAS LECCIONES ENTRE LOS EQUIPOS

La información que surge durante una retrospectiva de Sprint es casi siempre tremendamente valiosa. ¿Tiene este equipo problemas de concentración porque los gerentes de ventas insisten en secuestrar programadores para participar como “expertos técnicos” en reuniones de ventas? Esta es una información importante. Quizás otros equipos tengan los mismos problemas. ¿Se debería estar formando más a los responsables de producto acerca de los productos, de forma que puedan hacer el soporte a ventas ellos mismos?

Una retrospectiva de Sprint no trata sólo de cómo este equipo puede hacerlo mejor el próximo Sprint, tiene implicaciones más amplias que esa. La estrategia para manejar este hecho es muy simple. Una persona atiende a todas las reuniones de retrospectiva y actúa como puente de conocimiento. Muy informal. Una alternativa sería que cada equipo Scrum publique un informe de la retrospectiva de Sprint. Lo cual se ha intentado, pero encontramos que no mucha gente lee esos informes, y muchos menos actúan basándose en ellos. Así que se hace de la forma simple.

Reglas importantes para la persona que actúa como “puente de conocimiento”:

- Debería ser bueno escuchando.
- Si la retrospectiva es poco activa, se debería estar listo para realizar preguntas simples pero bien apuntadas para estimular la discusión dentro del grupo. Por ejemplo, “si se pudiera rebobinar y hacer este Sprint otra vez desde el día 1, ¿qué se haría de forma diferente?”
- Debe estar dispuesto a pasar tiempo visitando todas las retrospectivas de todos los equipos.
- Debería tener algún tipo de autoridad, de forma que pueda actuar sobre las sugerencias que estén fuera del control del propio equipo. Esto funciona bastante bien, pero puede que haya otras aproximaciones que funcionen mejor.

CÓMO SE HACEN LAS PRUEBAS EN SCRUM

Hacer pruebas es la parte probablemente que más variará entre diferentes organizaciones. Dependiendo de cuántos encargados de pruebas se cuente, cuanta automatización se tenga, con qué tipo de sistema cuente (¿simplemente servidor + aplicación Web, o de hecho vendes software en cajas?), tamaño de los ciclos de entrega, cuán crítico sea el software (servidor de blogs vs. sistema de control de vuelos), etc.

El encargado de pruebas es quien da el visto bueno. Además de ser solo un miembro del equipo, el encargado de pruebas tiene una labor importante. Es el que da el visto bueno. Nada se considera terminado hasta que él dice que está terminado. Se ha encontrado a muchos desarrolladores que dicen que algo está terminado cuando en realidad no lo estaba. Incluso si tienes una definición muy clara de lo que significa terminado, los desarrolladores frecuentemente la olvidarán.

Los programadores son gente impaciente y quieren dedicarse al próximo elemento lo antes posible.

Se ha experimentado muchísimo sobre cómo hacer pruebas en Scrum. En el mundo Scrum ideal, un Sprint produce una versión potencialmente instalable del sistema. Por ello, la implementación de planes de pruebas dentro y fuera del sprint es en todo caso opcional aun cuando el Scrum Team esté comprometido con la calidad. Debido a que algunas actividades del siguiente sprint pueden ser la solución de errores previos.⁵

2.4 MARCO CONCEPTUAL

- **Backoffice:** Un back office (oficina trasera) es la parte de las empresas donde se realizan las tareas destinadas a gestionar la propia empresa y con las cuales el cliente no necesita contacto directo. Por ejemplo: el departamento de informática y comunicaciones, el departamento de recursos humanos, el de contabilidad, etc. Conocido también como sistemas de apoyo al negocio donde back office corresponde a "todo lo que no está frente al cliente".
- **Elemento del product backlog.** En Scrum, un elemento del backlog del producto ("PBI", "elemento o ítem del backlog") es una unidad de trabajo lo suficientemente pequeña para que el equipo pueda completarla en un sprint (iteración).
- **Empowerment.** Delegación de atribuciones.

⁵ Itzcoalt, A. M, (2010). Desarrollo ágil con Scrum. Joiz.Net

- **Entrega.** Una entrega es la transición de un incremento potencialmente productivo del producto en algo que los clientes usen rutinariamente. Las entregas suelen ocurrir cuando uno o más sprints resultan en que el producto tiene suficiente valor como para superar el costo de desplegarlo.
- **Gráfico de burndown.** Los gráficos de burndown muestran el trabajo restante en el tiempo. El trabajo restante es el eje Y y el tiempo en días es el eje X. El trabajo restante debería subir y bajar y eventualmente tener una tendencia descendente.
- **Gráfico de burndown de la entrega.** Brinda una visión general sobre el progreso de la entrega. Muestra cuánto trabajo queda hacer al principio de cada sprint, para una única entrega. El alcance de este gráfico es una única entrega; sin embargo, un gráfico de burndown del producto abarca todas las entregas.
- **Gráfico de burndown del producto.** Es en donde se puede ver el progreso mensual (por sprint). Muestra cuánto trabajo restante hay al comienzo de cada sprint.
- **Gráfico de burndown del sprint.** Es el lugar donde se puede ver el progreso diario. Muestra dónde está el equipo respecto a completar las tareas correspondientes a los elementos del backlog del producto que cumplen el objetivo del sprint.
- **Impedimentos.** Cualquier cosa que le impida al equipo desempeñarse lo más eficientemente posible. Cada miembro del equipo puede anunciar un impedimento durante la reunión diaria de Scrum. El Scrum Master está a cargo de resolver los impedimentos.
- **Product backlog.** Es la lista de requerimientos del producto, llamados historias. Ordenados por prioridad y con un valor cuantificado en puntos.
- **Product Owner.** Rol de la metodología Scrum. Su papel es el de interactuar con el cliente. Es el encargado de crear y mantener el product backlog.
- **Punto.** Unidad utilizada en el product backlog para cuantificar el esfuerzo que supone el desarrollo de una historia.
- **Roles de scrum.** Existen tres roles en cualquier proyecto de Scrum: Product Owner, Scrum Team y Scrum Master.
- **Scrum.** Metodología de desarrollo de software ágil. Se compone de un seguido de iteraciones (sprints) en las cuales se va desarrollando el producto. Su composición está definida por un conjunto de roles.
- **Scrum Master.** Rol de la metodología Scrum. Su papel es el de velar por el correcto uso y ejecución del método. Además, se encarga de que el Scrum

Team no tenga interrupciones externas y, a su vez, transmite la voz de éste al exterior.

- **Sprint.** Iteración del método Scrum. Su duración puede variar pero es siempre fija durante una implantación del método. En ella se desarrolla el producto.
- **Sprint backlog.** Lista de historias a realizar durante el sprint. Divididas en tareas con un valor cuantificado en horas.
- **Stakeholder.** Persona relacionada con un proyecto pero sin estar implicada en su desarrollo.
- **Tarea.** Subdivisión de una historia. El sprint backlog está compuesto de tareas y cada una de ellas tiene asignado un tiempo estimado de desarrollo. Indican y describen el trabajo a realizar.
- **Scrum Team.** Rol de la metodología Scrum. Encargados del desarrollo del producto. Está compuesto generalmente por un total de 7 ± 2 miembros y se trata de un equipo multidisciplinario, con capacidad de auto-gestión y auto-organización.
- **Velocidad.** En Scrum, la velocidad es cuánto esfuerzo del backlog del producto el equipo puede manejar en un sprint. Esto puede estimarse viendo los sprints pasados, asumiendo que la composición del equipo y la duración del sprint se mantienen constante. También puede establecerse sprint-a-sprint, usando una planificación basada en compromisos.

En este capítulo se pudo conocer el Marco de Referencia, que involucra los antecedentes referidos a esta tesis donde se vio que soluciones se dieron en otras organizaciones ante una problemática similar; la metodología que se implantó y que resultados se obtuvo. Además se da a conocer la base teórica sobre la cual se sustenta la metodología Scrum, el modelo aplicativo que nos da una secuencia de fases a seguir y por último el marco conceptual que hace referencia a conceptos claves para Scrum.

CAPÍTULO III

INTERVENCIÓN METODOLÓGICA

En este capítulo se aplicará la Metodología de Desarrollo Ágil Scrum a un proyecto real fase por fase, mediante la aplicación de la misma se pretende identificar las verdaderas necesidades del cliente, realizar una correcta estimación de los tiempos, lograr una participación activa del equipo de trabajo, desarrollar un incremento funcional en el producto, revisar lo construido contrastando con la meta del sprint para entregar una versión del producto hasta finalmente obtener el producto esperado, lo cual no implica el fin del proyecto porque se deberá hacer un mantenimiento para permitir la continuidad del producto.

3.1. FASE N° 1: DEFINICIÓN DEL BACKLOG DEL PRODUCTO

Para el presente estudio se ha determinado realizar la intervención metodológica en el proyecto denominado “Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.”. El cliente solicitante del proyecto es La Positiva Seguros S.A., representado por el Sr. Jorge Shimizu Jeri (Apoderado de Ingeniería de Procesos).

A. DESCRIPCIÓN DEL PROYECTO

La Positiva Seguros S.A. cuenta con diversos tipos de seguros, entre ellos el seguro vehicular.

Actualmente el proceso de decisión de la cobertura vehicular se realiza en un documento excel, tal como se muestra en el gráfico N° 3.1.

Del gráfico de la página siguiente, se observa la hoja “Cotizador Vehiculos” del libro excel proporcionado por el cliente. Donde se muestra la interfaz utilizada actualmente para establecer la cobertura vehicular tras el ingreso de las características particulares del vehículo solicitante del seguro vehicular. Sin

embargo, esta forma de trabajo se hace tediosa en cuanto al despliegue de este documento a cada computadora de usuario final y a la actualización de data en tiempo real a una base de datos de Microsoft SQL Server 2008. Debido a ello, el dueño del producto desea que se automatice este proceso mediante una aplicación cliente.

Gráfico N° 3. 1
Interfaz del Cotizador vehicular actual en Excel – La Positiva Seguros S.A.

La Positiva Seguros
Esq. Javier Prado (Este) y Fco. Masías 370, San Isidro, Lima-Perú
Teléfono: 2110000 - Fax: 2110011

COTIZACIÓN DE SEGURO VEHICULAR
Seguro de Daños Propios

Cotización N°:
Fecha: *****

Uso: Transporte d
Oficina: Lima
Producto: Daño Propio
Plan R.C.:

DATOS DEL CONTRATANTE

Ap. Paterao / Razón Social: Ap. Materao: Nombres: Documento:
SR. DNI

DATOS DEL(OS) VEHÍCULO(S)

N°	Clase:	Marca:	Modelo:	Año:	Timón	Suma	Requiere	Tasa %:	Prima Neta (US\$):
					Cambiado Asegurado:	GPS:			
1					SI				
2									
3									
4									
5									
6									

Inicio Base de Datos Cotizador Vehiculos Solicitud de Emisión Parametros Actualizaciones Pendientes

Fuente: Área de Ingeniería de Procesos – La Positiva Seguros S.A.

Elaboración: La Positiva Seguros S.A.

La finalidad del proyecto es obtener un tarifario de cobertura vehicular en base a reglas de negocio, dadas por La Positiva Seguros S.A., mediante la creación de lenguaje de reglas con la tecnología Inrule 4.5 y la elaboración de dos aplicativos: un aplicativo cliente en Visual Studio .Net C# para el cálculo de la prima y un aplicativo cliente en Visual Basic 6.0 para la aplicación de reglas.

El principal objetivo del proyecto es:

- Automatizar el proceso de establecimiento de la cobertura vehicular utilizando el gestor de reglas de negocio Inrule Technology.

Los objetivos secundarios que persigue el proyecto son:

- Verificar la capacidad de Inrule Technology para procesos de carga masiva de datos.

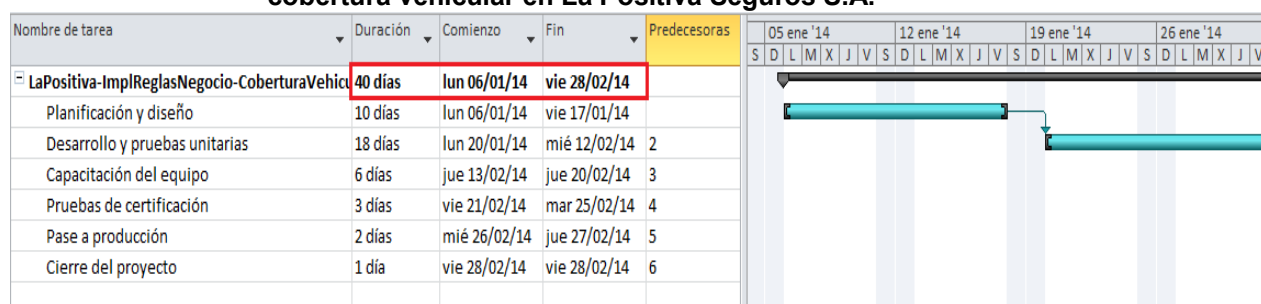
- Verificar la flexibilidad de Inrule Technology frente a cambios en los procesos de negocio.

El proyecto llegó a la unidad de negocio de TI a inicios del mes de enero del año 2014. El mismo se desarrolló involucrando las siguientes tecnologías:

- InRule Technology 4.5
- Visual Studio .Net 2012
- Visual Basic 6.0
- Microsoft SQL Server 2008 R2
- Internet Information Services 7.5
- SOAP Toolkit 3.0

En el gráfico N° 3.2 se observa el cronograma del proyecto “Implementación de las reglas de negocio para establecer la cobertura vehicular en la Positiva Seguros S.A.”.

Gráfico N° 3. 2
Cronograma del proyecto “Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.”



Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

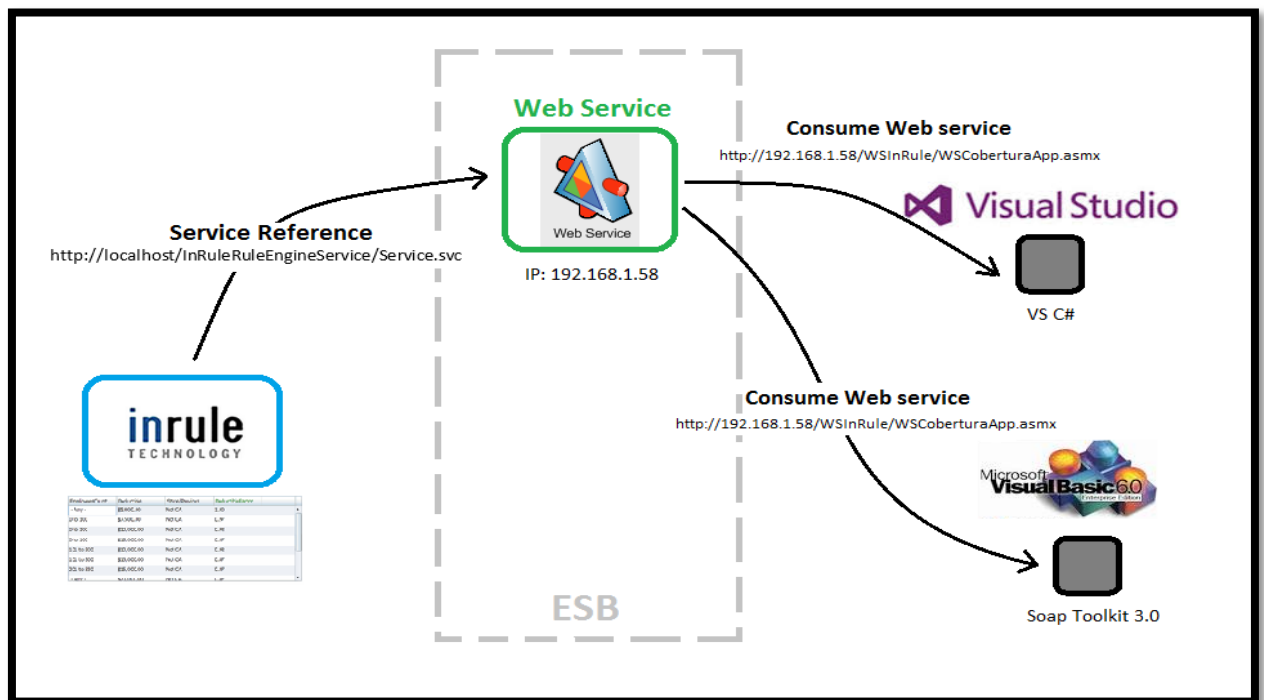
Elaboración: Propia

Del gráfico, se observa que según el cronograma el proyecto inicia el lunes 06/01/2014 y finaliza el viernes 28/02/2014, con una duración de 40 días de trabajo, y con un total de 4 recursos humanos asignados al proyecto.

Este proyecto consiste en la creación de un lenguaje de reglas de negocio basado en la base de datos excel “Cotizador_vehicular.xls” proporcionada por el cliente. El mismo será construido íntegramente utilizando la herramienta Inrule Technology 4.5, que es un Sistema de Gestión de Reglas de Negocio (BRMS: Business Rules Management System) y se desarrollarán dos formularios del lado del usuario final en Visual Studio .Net 2012 C# y Visual Basic 6.0.

En el gráfico N° 3.3 se visualiza el esquema general de la arquitectura del proyecto con las respectivas aplicaciones involucradas.

Gráfico N° 3. 3
Esquema general de la arquitectura del proyecto



Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico se visualiza la arquitectura del proyecto, descrita a continuación:

- Al lado izquierdo se tiene el aplicativo Inrule (Nombre: CoberturaVehiculo_TD.ruleapp), el cual contiene las entidades, tablas de decisiones y reglas de negocio.
- En el centro, se tiene el servicio web (Nombre: WSCobertura), el cual hace uso de una referencia al servicio de Inrule, y que será consumido por los dos aplicativos.
- Y al lado derecho se tienen los dos aplicativos, tanto en Visual Studio .net con C# (Nombre: App_InRule_NET) y en Visual Basic 6.0 (App_InRule_VB6), los cuales son interfaces funcionales de cara al usuario final.

B. DEFINIENDO LA PILA DE PRODUCTO

En esta primera fase de aplicación de la metodología Scrum, se definirá el product backlog, que es básicamente una lista de requerimientos de usuario priorizada y proporcionada por el dueño del producto, tal como se muestra en la tabla N° 3.1.

Tabla N° 3. 1
Product Backlog del proyecto proporcionado por el dueño del producto

PRODUCT BACKLOG			
ID	Descripción de requerimiento	Importancia	Notas
1	Nuevo formulario de Ingreso al Sistema Cobertura Vehicular en Visual Studio .Net 2012	20	
2	Ingreso de datos de entrada de características del vehículo	50	
3	Aplicación de las reglas de negocio en el aplicativo .Net mediante el botón “Calcula prima”	High (Alto)	
4	Consulta y verificación de resultados de la cobertura vehicular por el usuario final	80	
5	Nuevo formulario de Ingreso al Sistema Cobertura Vehicular en Visual Basic 6.0	20	
6	Ingreso de datos de entrada de características del vehículo	50	
7	Aplicación de las reglas de negocio en el aplicativo VB6.0 mediante el botón “Apply rules”	High (Alto)	
8	Consulta y verificación de resultados de la cobertura vehicular por el usuario final	80	

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla, se puede notar que el product backlog posee términos para un cliente que tiene cierto dominio técnico del tema. La definición de este listado es un punto crucial en el proceso porque permitirá determinar posteriormente los sprints para estimar correctamente los tiempos de desarrollo del proyecto.

Usualmente el product backlog se elabora en un documento excel, en este caso además contiene la data para el proyecto, con compartición permitida, tal como se muestra en el gráfico N° 3.4.

Gráfico N° 3. 4
Base de datos del cliente del proyecto Inrule Technology

COBERTURA												
USO		PARTICULAR						URBANO			ALQUILER	
CLASIFICACION		Vehículos Livianos						Vehículos Menores	Vehículos Livianos (Taxi)		Vehículos Pesados (Público)	Vehículos Livian-
CLASE		Automovil, Camioneta Station Wagon, Camioneta Panel, Camioneta Rural				Camioneta Pickup		Motocicletas	Automovil, Camioneta Station Wagon, Camioneta Rural		Ómnibus y Microbus	Automovil, Camioneta Station Wagon, Camioneta Panel, Camioneta Rural, Camioneta Picku
PROCEDENCIA		No Chino, No Indio, No Timón Cambiado			Chino, Indio, Timón Cambiado	No Chino, No Indio, No Timón Cambiado	Chino, Indio, Timón Cambiado	Cualquier procedencia	No Chino, No Indio, No Timón Cambiado	Chino, Indio	Timón Cambiado	Cualquier procedencia
COBERTURAS PRINCIPALES		Bajo Riesgo	Mediano Riesgo	Alto Riesgo	Chino, Indio, Timón Cambiado	No Chino, No Indio, No Timón Cambiado	Chino, Indio, Timón Cambiado	Cualquier procedencia	No Chino, No Indio, No Timón Cambiado	Chino, Indio	Timón Cambiado	Cualquier procedencia
Daño Propio (choque, vuelco, incendio, robo total y/o parcial, rotura de lunas), Pérdida Total, hasta		Valor Asegurado			Valor Comercial	Valor Asegurado	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial
Responsabilidad Civil frente a Terceros, hasta		US\$100,000			US\$100,000	US\$100,000	US\$100,000	US\$100,000	US\$100,000	US\$100,000	US\$100,000	US\$100,000
Responsabilidad Civil frente a Ocupantes, (según tarjeta de propiedad, excepto Accidentes Personales de Ocupantes (según tarjeta de propiedad))		Máximo 5			Máximo 5	Máximo 5	Máximo 5	Máximo 5	No Cubre	No Cubre	No Cubre	Máxim
Muerte e Invalidez Permanente, hasta		US\$20,000 o/u			US\$20,000 o/u	US\$20,000 o/u	US\$20,000 o/u	US\$20,000 o/u	US\$20,000	US\$20,000	US\$20,000	US\$20,000
Gastos de Curación, hasta		US\$4,000 o/u			US\$4,000 o/u	US\$4,000 o/u	US\$4,000 o/u	US\$4,000 o/u	US\$4,000	US\$4,000	US\$4,000	US\$4,000
Gastos de Sepelio, hasta		US\$1,000 o/u			US\$1,000 o/u	US\$1,000 o/u	US\$1,000 o/u	US\$1,000 o/u	US\$1,000	US\$1,000	US\$1,000	US\$1,000
COBERTURAS ADICIONALES		Valor Asegurado			Valor Comercial	Valor Asegurado	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial
Riesgos de la Naturaleza, hasta		Valor Asegurado			Valor Comercial	Valor Asegurado	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial
Uso de Vías No Autorizadas, hasta		Valor Asegurado			Valor Comercial	Valor Asegurado	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial
Huelga, Comoción Civil, Daño Malicioso, Vandalismo y Terrorismo, hasta		Valor Asegurado			Valor Comercial	Valor Asegurado	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial	Valor Comercial
Cobrador Vehículos Solicita de Emisión Parametros ActualizacionesPendientes Tarifario Base Coberturas Base Tasas Base Modelos												

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico, para el proyecto los desarrolladores tuvieron acceso y permisos de lectura y escritura al documento excel “Cotizador_vehicular.xls” para poder visualizar y clarificar la lista de requerimientos de usuario (para mayor detalle referirse al Anexo III).

3.2. FASE N° 2: PLANIFICACIÓN DEL SPRINT

Para llevar a cabo la reunión de planificación de sprint, previamente el equipo debió asegurarse que el product backlog se encuentre bien definido.

El equipo para este proyecto fue conformado de la siguiente manera:

Product Owner: Jorge Shimizu Jeri

Scrum Master: Ing. Moshe Espinoza Chueca Ruiz

Scrum Team:

- Lizbet Velásquez Q.
- James Mendoza S.
- Carlos Malpica V.

Siendo los comprometidos: el product owner, el scrum team y el scrum master. Y los implicados; los usuarios finales, el Área de Procesos y el Área Comercial.

La primera reunión de planificación de sprints, permitirá que el equipo Scrum estructure los sprints necesarios, además que realice todas las estimaciones iniciales y que verifique las importancias establecidas por el cliente, tal como se muestra en la tabla N° 3.2.

Tabla N° 3. 2

Product Backlog del proyecto conteniendo la Importancia y la Estimación inicial

PRODUCT BACKLOG					
ID	Nombre	Importancia	Estimación inicial	Cómo probarlo	Notas
1	Ingreso al Sistema Cobertura Vehicular en Visual Studio C# .Net 2012	20	5	Hacer doble clic sobre el ícono del programa	Según Análisis funcional del proyecto
2	Ingreso al Sistema Cobertura Vehicular en Visual Basic 6.0	20	5	Hacer doble clic sobre el ícono del programa	Según Análisis funcional del proyecto
3	Aplicación de las reglas en el aplicativo Visual Studio .Net	High (Alto)	8	Hacer clic sobre el botón Calcula Prima	Se necesita un web service para ser consumido
4	Aplicación de las reglas en el aplicativo Visual Basic 6.0	High (Alto)	7	Hacer clic sobre el botón Apply Rules	Se necesita un web service para ser consumido
5	Consumir el servicio web desde el aplicativo Visual Studio .Net	50	5	Debe devolver una acción de acuerdo a las condiciones ingresadas	Verificar regla de negocio con documento excel proporcionado
6	Consumir el servicio web desde el aplicativo Visual Basic 6.0	50	5	Debe devolver una acción de acuerdo a las condiciones ingresadas	Verificar regla de negocio con documento excel proporcionado
7	Consulta y verificación de resultados según la lógica del negocio (VS .Net)	80	3	Contrastar resultados con archivo Excel o tabla de decisiones	Verificar según documentación de la lógica del negocio
8	Consulta y verificación de resultados según la lógica del negocio (VB 6.0)	80	2	Contrastar resultados con archivo Excel o tabla de decisiones	Verificar según documentación de la lógica del negocio

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

La primera reunión se realizó mediante el siguiente orden:

Primera reunión de planificación de Sprint (SPRINT 1):

Fecha: Lunes 06/01/2014

Hora: 9:00a.m. – 11:00a.m.

Lugar: Instalaciones de La Positiva Seguros S.A. - Piso 13 Sala “C”

Próxima reunión: Lunes 20/01/2014

- ✓ **9:00 – 9:30.** El dueño de producto comenta la meta del sprint y resume la Pila de Producto. Se establece el lugar, fecha y hora para la revisión del sprint.

Meta de primer sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos
- ✓ **9:30 – 10:00.** El equipo Scrum da estimaciones de tiempo, y divide los elementos tanto como sea necesario de acuerdo a su experiencia. El dueño de producto actualiza los ratios de importancia. Se clarifican los elementos. Para todos los elementos de alta importancia se establece la columna “Cómo probarlo”.
- ✓ **10:00 – 10:30.** El equipo selecciona las historias que se incluirán en el Sprint. Se realizan cálculos de velocidad para chequear si es factible.
- ✓ **10:30 – 11:00.** Se selecciona un lugar y hora para el Scrum Diario. Se continúa dividiendo las historias en tareas.

El “sprint planning” es una reunión crítica, probablemente el evento más importante en Scrum, ya que una reunión de planificación mal ejecutada puede llevar a incumplir un sprint entero.

El propósito de la reunión de planificación de sprint es dar al equipo suficiente información para ser capaz de trabajar en paz por dos semanas, y proporcionarle al dueño de producto los entregables de la meta de sprint en la fecha acordada para su revisión y retrospectiva de ser el caso.

Para realizar la estimación del tiempo total empleado en el proyecto, el Scrum Master junto al Equipo Scrum deben evidenciar todo los posibles inconvenientes o circunstancias que pueden ocurrir. En el proyecto se tuvieron las siguientes consideraciones: Uno de los desarrolladores, en este caso James (Desarrollador 1) saldrá de vacaciones 15 días por lo cual sólo apoyará al proyecto 10 días. Carlos (Desarrollador 2) apoyará al proyecto 10 días por estar asignado a otro proyecto a tiempo parcial. Y Lizbet (Desarrollador 3) estará enfocada en el desarrollo por 20 días y será la líder del equipo Scrum. Consolidando todo ello tenemos:

Días Disponibles

James	10
Lizbet	20
Carlos	10

40 DÍAS-HOMBRE disponibles

Lo cual da un total de 40 días- hombres disponibles para el proyecto. Ésta estimación se ve reflejada en el cronograma de la propuesta técnica proporcionada al cliente.

La tabla N° 3.3 muestra los tres sprint backlog definidos en la primera reunión de planificación de sprints.

Tabla N° 3. 3

Sprints backlog definidos en la primera reunión de planificación

Item	Sprint	Responsable(s)	Tareas	Días asignados
1	SPRINT 1	Desarrollador 1 Desarrollador 3	✓ Realizar el Análisis y Diseño del proyecto	10
			✓ Realizar el Modelamiento de la Base de datos	
2	SPRINT 2	Desarrollador 1 Desarrollador 3	✓ Creación de entidades, campos calculados y carga de data en tablas de decisión en InRule	15
			✓ Creación de la lógica de negocio mediante el lenguaje de reglas en InRule	
3	SPRINT 3	Desarrollador 2 Desarrollador 3	✓ Creación de web service en Visual Studio .Net C#	15
			✓ Creación de los dos aplicativos: un aplicativo en Visual Studio .Net y otro aplicativo en Visual Basic 6.0	

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

La tabla contiene los responsables por cada tarea de cada sprint, que fue producto de la primera reunión de planificación de sprints. Donde el Sprint 1 tendrá una duración de 10 días (2 semanas laborales), el Sprint 2 tendrá una duración de 15 días (3 semanas laborales) y el Sprint 3 tendrá una duración de 15 días (3 semanas laborales).

La estimación de tiempo para el Sprint 1, se determinó de la siguiente manera:

Días Disponibles

James	5
Lizbet	5

10 DIAS-HOMBRE disponibles

7
3

$V_{estimada} = 10$

Para el proyecto, considerando que el factor de dedicación para este primer sprint es igual a: $F_{\text{dedicación}} = 10/10 = 1$. Entonces, la velocidad estimada será igual a: $V_{\text{estimada}} = 10 \times 1 = 10$.

La segunda reunión de planificación del sprint se realizó mediante el siguiente orden:

Segunda reunión de planificación de Sprint (SPRINT 2):

Fecha: Lunes 20/01/2014

Hora: 9:00a.m. – 11:00a.m.

Lugar: Instalaciones de La Positiva Seguros S.A. - Piso 13 Sala “C”

Próxima reunión: Lunes 10/02/2014

- ✓ **9:00 – 9:30.** El equipo Scrum proporciona al dueño del producto los entregables de producto de las metas de sprint acordadas en la reunión anterior.
- ✓ **9:30 – 10:00.** El dueño de producto verifica las metas de sprint y valida si es lo que solicitó en el product backlog. En este caso hubo un error en el modelamiento de la base de datos y se deberá hacer la respectiva retrospectiva del sprint.
- ✓ **10:00 – 10:30.** El dueño de producto establece la meta del sprint para el siguiente sprint. Se establece el lugar, fecha y hora para la revisión del sprint. El equipo selecciona las historias que se incluirán en el sprint.

Meta de primer sprint:

- ✓ Creación de entidades, campos calculados y carga de data en tablas de decisión en InRule
- ✓ Creación de la lógica de negocio mediante el lenguaje de reglas en InRule
- ✓ **10:30 – 11:00.** Se selecciona un lugar y hora para el Scrum Diario. El equipo Scrum continúa dividiendo las historias en tareas.

La estimación de tiempo para el Sprint 2, se determinó de la siguiente manera:

Días Disponibles		<div><div>8</div><div>7</div></div> <div>$V_{estimada} = 15$</div>
James	5	
Lizbet	10	
<hr/>		
15 DIAS-HOMBRE disponibles		

Para el proyecto, la velocidad estimada será igual a: $V_{estimada} = 15 \times 1 = 15$, considerando que el factor de dedicación es igual a 1.

La tercera y última reunión de planificación del sprint se realizó mediante el siguiente orden:

Tercera reunión de planificación de Sprint (SPRINT 3):

Fecha: Lunes 10/02/2014

Hora: 9:00a.m. – 11:00a.m.

Lugar: Instalaciones de La Positiva Seguros S.A. - Piso 13 Sala “C”

Próxima reunión: Viernes 28/02/2014

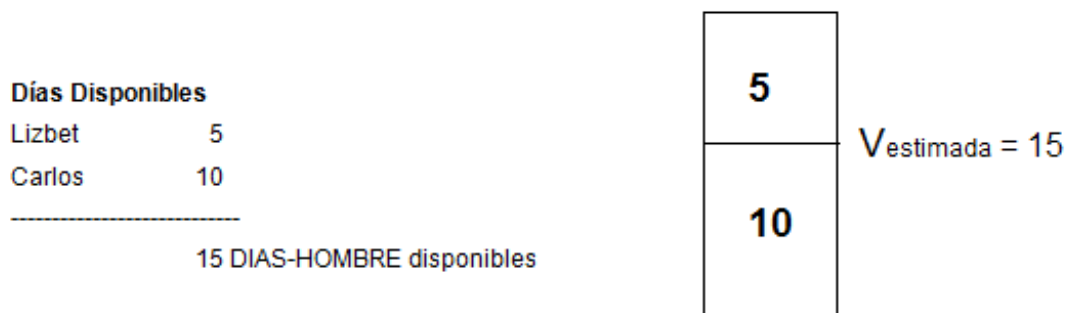
- ✓ **9:00 – 9:30.** El equipo Scrum proporciona al dueño del producto los entregables de producto de las metas de sprint acordadas en la reunión anterior.
- ✓ **9:30 – 10:00.** El dueño de producto verifica las metas de sprint y valida si es lo que solicitó en el product backlog. En este caso no existieron errores y el dueño de producto dio su aprobación.
- ✓ **10:00 – 10:30.** El dueño de producto establece la meta del sprint para el siguiente sprint. Se establece el lugar, fecha y hora para la revisión del sprint. El equipo selecciona las historias que se incluirán en el sprint.

Meta de primer sprint:

- ✓ Creación de web service en Visual Studio .Net C#
- ✓ Creación de los dos aplicativos: un aplicativo en Visual Studio .Net y otro aplicativo en Visual Basic 6.0

- ✓ **10:00 – 10:30.** El Scrum Master coordina la Integración del aplicativo, la realización de pruebas y el respectivo cierre del proyecto con el dueño del producto y el equipo Scrum.

La estimación de tiempo para el Sprint 3, se determinó de la siguiente manera:



Para el proyecto, la velocidad estimada será igual a: $V_{estimada} = 15 \times 1 = 15$, considerando que el factor de dedicación es igual a 1.

3.3. FASE N° 3: SCRUM DIARIO

A. COMUNICACIÓN DE SPRINT BACKLOGS

Para poder comunicar el avance de cada uno de los tres sprints backlogs, se realizan los scrum diarios o reuniones diarias, en donde participan el Scrum Master y el Equipo Scrum principalmente para verificar y evaluar el avance realizado por los responsables de las tareas asignadas. La finalidad de ello es que ninguna tarea sea un cuello de botella que impida la culminación del proyecto.

Sobre una gran pizarra, y con la ayuda de post-its y plumones, se construyó la tabla de tareas para el proyecto y se comunicaron los avances de los sprints backlogs (sprint 1, sprint 2 y sprint 3). Tal como se muestra en el gráfico N° 3.5.

Del gráfico de la página siguiente, se observa que es importante la comunicación de las tareas que se vienen realizando para ver el comportamiento del gráfico burndown a lo largo del proyecto, y tener así la perspectiva de si el avance es óptimo (para mayor detalle referirse al Anexo IV).

Gráfico N° 3. 5
Sprint backlog para el proyecto Inrule Technology

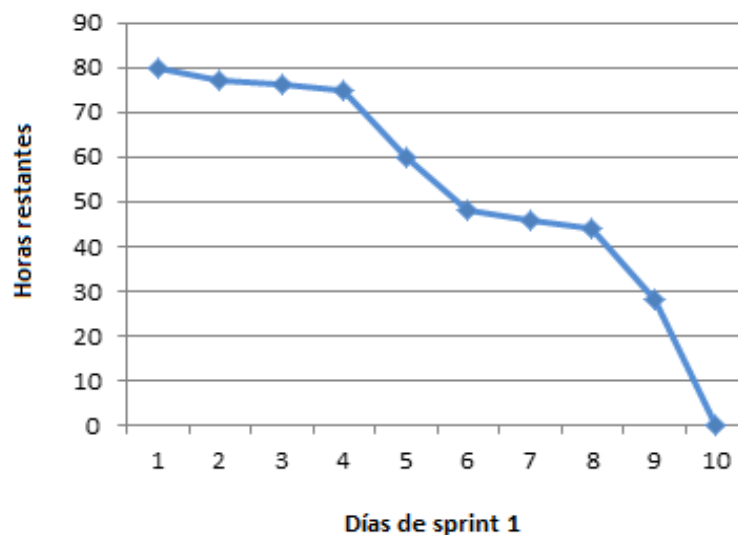


Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

B. TRABAJANDO CON EL CUADRO BURNDOWN

El gráfico N° 3.6 muestra el cuadro burndown o gráfica de progreso para el sprint backlog 1, el cual tuvo una duración de 10 días.

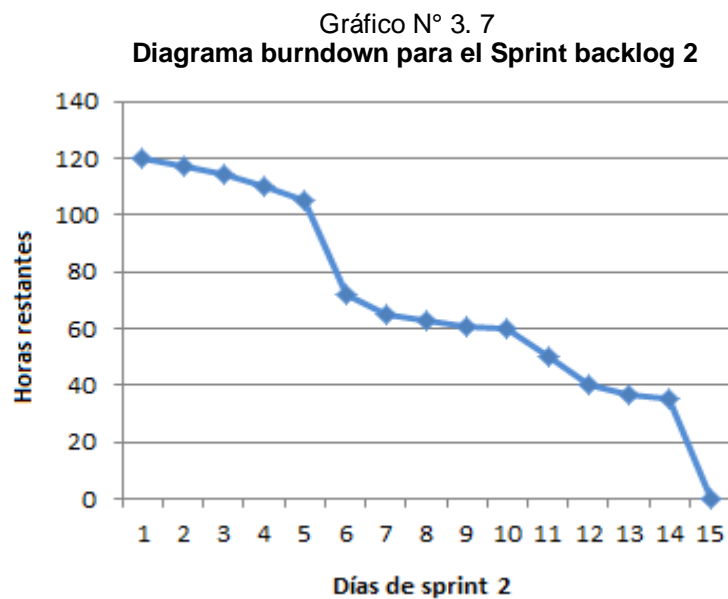
Gráfico N° 3. 6
Diagrama burndown para el Sprint backlog 1



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se puede observar el comportamiento de la gráfica lineal, la cual tiende al decrecimiento, ello debido a las reuniones diarias que permiten la evaluación de avances y la minimización de retrasos. De esta manera se va culminando el sprint 1 hasta pasar al siguiente sprint.

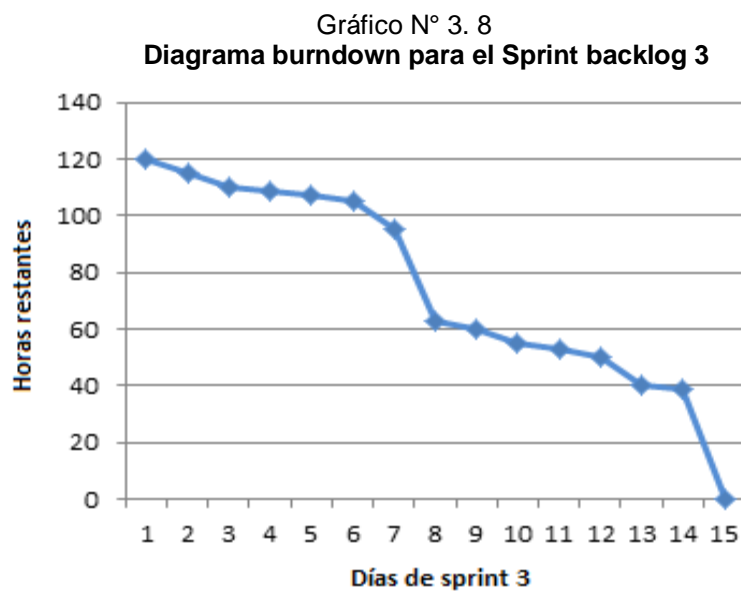
El gráfico N° 3.7 muestra el cuadro burndown o gráfica de progreso para el sprint backlog 2, el cual tuvo una duración de 15 días.



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se puede observar el comportamiento de la gráfica lineal, la cual tiende al decrecimiento.

El gráfico N° 3.8 muestra el cuadro burndown o gráfica de progreso para el sprint backlog 3, el cual tuvo una duración de 15 días.



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se puede observar el comportamiento de la gráfica lineal, la cual tiende al decrecimiento.

3.4. FASE N° 4: REVISIÓN DEL SPRINT

A. PLANIFICACIÓN DE ENTREGAS

Los entregables de cada sprint, se basan inicialmente en el product backlog definido en la primera etapa de la metodología, la Definición del Product Backlog. Asimismo se basan en las tareas establecidas en el Sprint Backlog, definidas en la segunda etapa de Planificación de Sprints.

Para el Sprint 1, se tenían las siguientes metas de sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos

La primera meta del Sprint 1, realizar el análisis y diseño del proyecto fue concluida en su totalidad sin mayor inconveniente, y en cuanto a la segunda meta del Sprint 1; inicialmente se cometió un error en el modelamiento de la base de datos de nombre InruleDB, el cual fue corregido en quinta etapa de Retrospectiva del Sprint.

El gráfico N° 3.9 muestra el diagrama de base de datos producto del modelamiento de datos inicial, realizado en Microsoft SQL Server 2008 R2.

Gráfico N° 3. 9
Modelo de datos de la base de datos InruleDB conteniendo un error



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

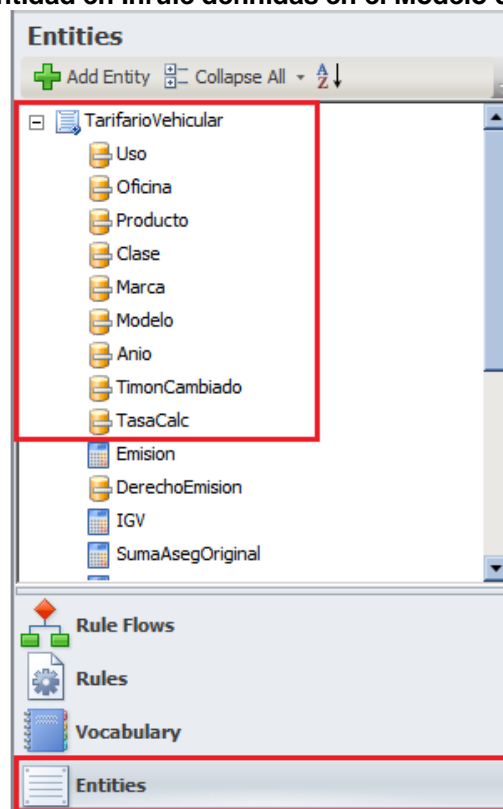
El gráfico de la página anterior muestra el resultado del modelamiento de datos, el cual fue mal entendido respecto a lo que el cliente requería porque se obvió la tabla donde se almacenan las oficinas “TbOficina”, de La Positiva Seguros a nivel nacional. Este error fue subsanado en la quinta etapa de la metodología Scrum, la Retrospectiva del Sprint.

Para el Sprint 2, se tenían las siguientes metas de sprint:

- ✓ Creación de entidades, campos calculados y carga de data en tablas de decisión en InRule
- ✓ Creación de la lógica de negocio mediante el lenguaje de reglas en InRule

Tanto la primera meta del Sprint 2 como la segunda meta del Sprint 2 se desarrollaron correctamente, tal como se muestra en el gráfico N° 3.10.

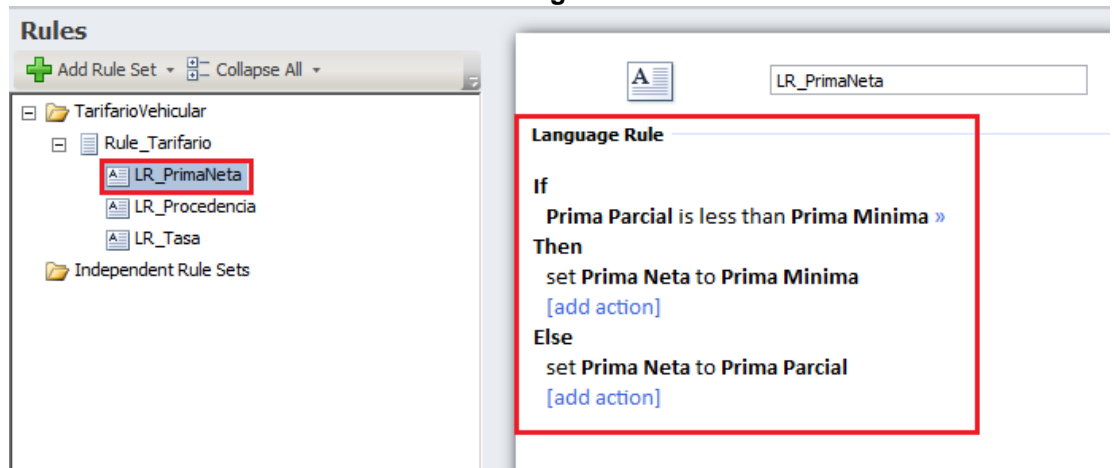
Gráfico N° 3. 10
Creación de Entidad en Inrule definidas en el Modelo de datos



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

El gráfico muestra la entidad principal creada en Inrule Technology, denominada “TarifarioVehicular”, la cual contiene sus campos (similar a los atributos de una tabla de base de datos), que serán los parámetros de entrada de la aplicación. Después, se establecen las condiciones a través del lenguaje de reglas, llamado “LR_PrimaNeta”. Tal como se muestra en el gráfico N° 3.11.

Gráfico N° 3. 11
Creación del Lenguaje de Reglas en Inrule según la lógica de negocio de La Positiva Seguros S.A.

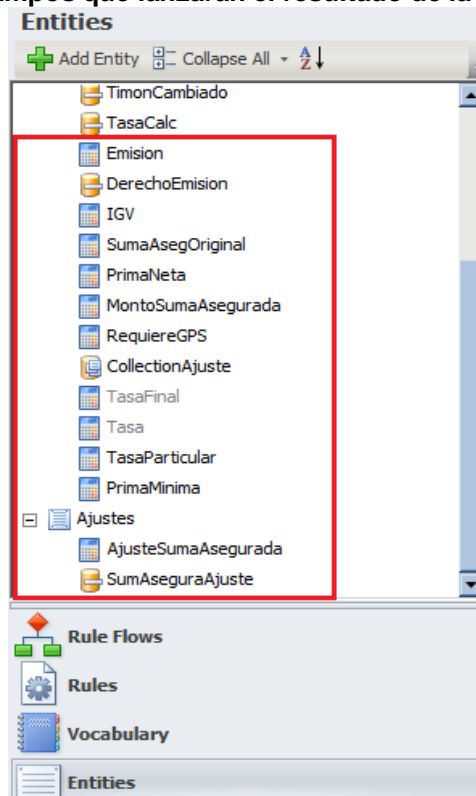


Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se visualiza la estructura condicional, de acuerdo a la lógica de negocio de La Positiva Seguros S.A.

Posteriormente se definen los campos que serán los parámetros de salida de la aplicación, tal como se muestra en el gráfico N° 3.12.

Gráfico N° 3. 12
Creación de los campos que lanzarán el resultado de la aplicación de la regla



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico de la página anterior se observa los campos solicitados por el cliente, los cuáles lanzan los resultados que permitirán establecer la cobertura vehicular de acuerdo a las características del vehículo ingresadas.

Una vez creada la estructura de la regla de negocio, se procede a llenar la data a través del empleo de elemento Inline Table propio de Inrule. Tal como se muestra en el gráfico N° 3.13.

Gráfico N° 3. 13
Carga de la data en Inrule a través de Inline Table

The screenshot shows the Inrule software interface. On the left, the 'Data' panel lists various tables: LstUso, LstOficina, LstProducto, LstClase, LstMarca, LstModelo, LstTiempo, LstAño, **TblTasa**, and TblModelos. The main area displays an 'Inline Table' for 'TblTasa'. The table has columns: Uso, Clasificación, Clase, Procedencia, Riesgo, Region, Año, Tasa, and PrimaNetoMinima. The table contains 10 rows of data for 'Particular' vehicles in Lima, ranging from 1993 to 2007.

Uso	Clasificación	Clase	Procedencia	Riesgo	Region	Año	Tasa	PrimaNetoMinima
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1993	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1994	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1995	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1996	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1997	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1998	6.90	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	1999	6.80	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2000	6.80	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2001	6.80	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2002	6.80	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2003	6.80	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2004	6.50	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2005	6.10	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2006	5.70	370
Particular	Vehículos Livianos	Automovil	Estandar	Bajo Riesgo	Lima	2007	5.40	370

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Las tablas cargadas con el elemento Inline Table son: TblTasa y TblModelos.

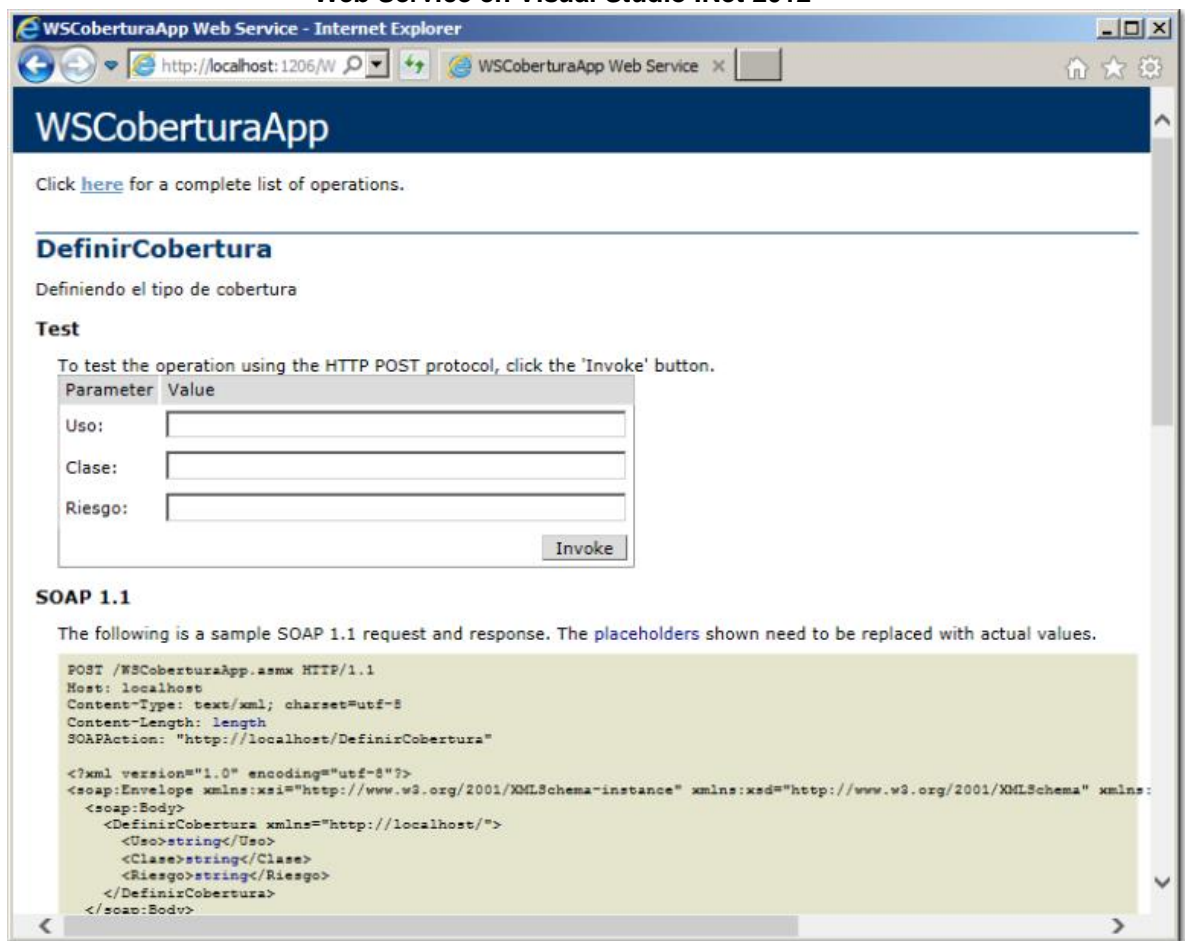
Para el Sprint 3, se tenían las siguientes metas de sprint:

- ✓ Creación de web service en Visual Studio .Net C#
- ✓ Creación de los dos aplicativos: un aplicativo en Visual Studio .Net y otro aplicativo en Visual Basic 6.0

Las metas definidas en el Sprint 3 se cumplieron en su totalidad, tal como se observa en el gráfico N° 3.14.

Del gráfico de la página siguiente se observa el servicio web desarrollado en Visual Studio .Net 2012 con el lenguaje C#.

Gráfico N° 3. 14
Web Service en Visual Studio .Net 2012



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

El gráfico N° 3.15 muestra el aplicativo en .Net, cuyo desarrollo fue considerado en el Sprint 3.

Del gráfico de la página siguiente se observa la interfaz del formulario desarrollado en Visual Studio .Net 2012 con el lenguaje de programación C#, la cual contiene el botón “Calcula prima”.

Gráfico N° 3. 15
Aplicativo en Visual Studio .Net 2012

COTIZACIÓN DE SEGURO VEHICULAR

Cotización N°: 100-1
 Fecha: 27/02/2014
 Nombres y Apellidos: JUAN PEREZ
 DNI: 44321811
 Uso: Particular
 Oficina: Lima
 Producto: Daño Propio
 Plan R.C.:

N° Clase:	Marca:	Modelo:	Año:	Timón cambiado:	Suma asegurada:	Requiere GPS:	Tasa %:	Prima (US\$):
Automovil	Alfa Romeo	145 1.4 TS 16V	1998	SI	4000	No	7.10	450

Calcula prima

Prima Neta (US\$): 450
 Derecho de Emisión (US\$): 8.52
 IGV (US\$): 55.58
 PRIMA TOTAL (US\$): 514.10
 PRIMA TOTAL (S/.): 514.10

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
 Elaboración: Propia

El gráfico N° 3.16 muestra el aplicativo en Visual Basic 6.0, cuyo desarrollo también fue considerado en el Sprint 3.

Gráfico N° 3. 16
Aplicativo en Visual Basic 6.0

Cobertura vehicular - La Positiva Seguros

Uso: Particular
 Clase: autos / station wagon / cmta. Panel y
 Riesgo: bajo riesgo

APPLY RULES

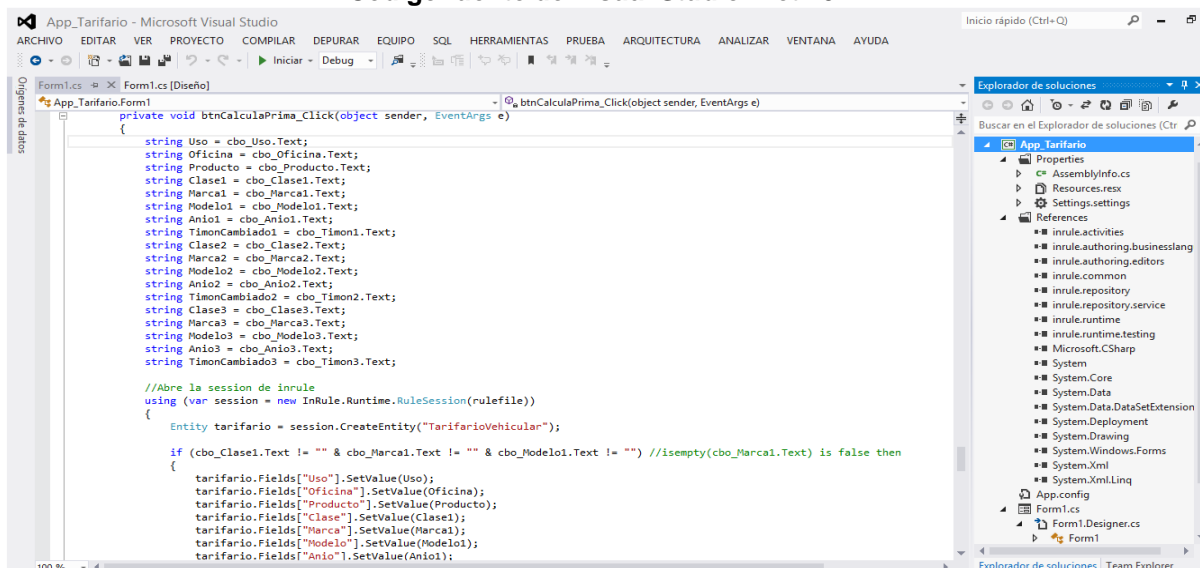
COBERTURA:

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
 Elaboración: Propia

Del gráfico se observa la interfaz del formulario desarrollado en Visual Basic 6.0, la cual contiene el botón “Apply Rules”.

El gráfico N° 3.17 muestra el código fuente trabajado para el aplicativo en Visual Studio .Net 2012, cuyo desarrollo fue considerado en el Sprint 3.

Gráfico N° 3. 17
Código fuente de Visual Studio .Net 2012



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico se observa el código desarrollado para el botón “Calcula prima”, el cual permitirá el cálculo de las tasas y primas (en soles y dólares) que el cliente adquiriente del seguro vehicular deberá pagar.

3.5. FASE N° 5: RETROSPECTIVA DEL SPRINT

En esta etapa se debe realizar la retrospectiva de cada uno de los tres sprints definidos en la fase de planificación de sprints, siempre y cuando el cliente y/o dueño de producto establezca que el entregable proporcionado por el equipo Scrum no es lo que se solicitó al inicio del proyecto.

En el presente estudio las retrospectivas para los Sprint 2 y Sprint 3 fueron satisfactorias. Por el contrario, la retrospectiva para el Sprint 1 no fue exitosa.

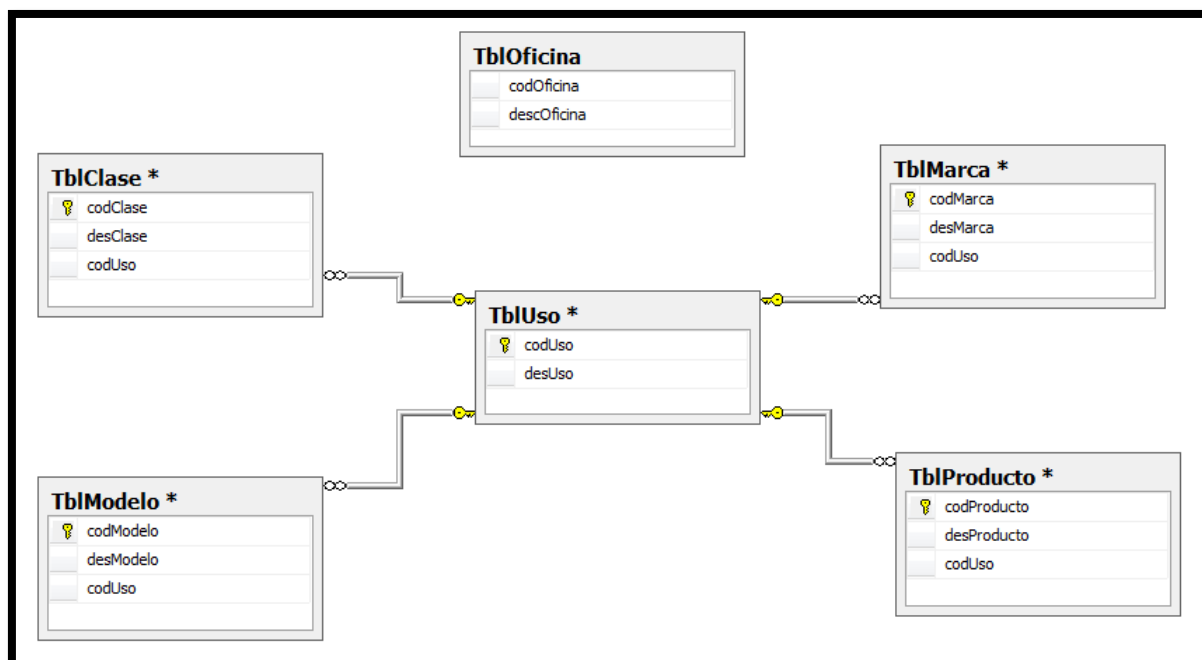
Para el Sprint 1, se tenían las siguientes metas de sprint:

- ✓ Realizar el Análisis y Diseño del proyecto
- ✓ Realizar el Modelamiento de la Base de datos

La primera meta del Sprint 1, realizar el análisis y diseño del proyecto fue concluida en su totalidad sin mayor inconveniente, y en cuanto a la segunda meta del Sprint 1; se cometió un error en el modelamiento de la base de datos de nombre InruleDB, al no considerarse la tabla TbIOficina. Dicha observación fue corregida en su totalidad en la etapa de retrospectiva del sprint 1.

El gráfico N° 3.18 muestra el modelo de datos de la base de datos InruleDB, modificado y corregido según las especificaciones del usuario en la segunda reunión de planificación de sprint.

Gráfico N° 3.18
Modelo de datos de la base de datos InruleDB con levantamiento del error



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Finalmente, la tabla N° 3.4 muestra el estado final de los ítems del product backlog.

Tabla N° 3. 4
Estado final de las tareas del Product Backlog del proyecto

PRODUCT BACKLOG					
ID	Nombre	Importancia	Estimación inicial	Cómo probarlo	Estado final
1	Ingreso al Sistema Cobertura Vehicular en Visual Studio C# .Net 2012	20	5	Hacer doble clic sobre el ícono del programa	REALIZADO
2	Ingreso al Sistema Cobertura Vehicular en Visual Basic 6.0	20	5	Hacer doble clic sobre el ícono del programa	REALIZADO
3	Aplicación de las reglas en el aplicativo Visual Studio .Net	High (Alto)	8	Hacer clic sobre el botón Calcula Prima	REALIZADO
4	Aplicación de las reglas en el aplicativo Visual	High (Alto)	7	Hacer clic sobre el botón Apply Rules	REALIZADO

	Basic 6.0				
5	Consumir el servicio web desde el aplicativo Visual Studio .Net	50	5	Debe devolver una acción de acuerdo a las condiciones ingresadas	REALIZADO
6	Consumir el servicio web desde el aplicativo Visual Basic 6.0	50	5	Debe devolver una acción de acuerdo a las condiciones ingresadas	REALIZADO
7	Consulta y verificación de resultados según la lógica del negocio (VS .Net)	80	3	Contrastar resultados con archivo Excel o tabla de decisiones	REALIZADO
8	Consulta y verificación de resultados según la lógica del negocio (VB 6.0)	80	2	Contrastar resultados con archivo Excel o tabla de decisiones	REALIZADO

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

La tabla refleja que la aplicación correcta de Scrum, permitió culminar cada historia de usuario planteada al inicio del proyecto.

Una vez levantadas cada una de las observaciones realizadas por el dueño del producto a los entregables desarrollados por el equipo Scrum, se procede a realizar el cierre del proyecto. Tarea que generalmente le corresponde al Scrum Master en coordinación con el dueño del producto.

La tabla N° 3.5 muestra el estado de las tareas definidas en la etapa de Planificación de Sprint.

Tabla N° 3. 5
Cierre del proyecto con Scrum

Item	Sprint	Responsable(s)	Tareas	Status	Status and Review
1	SPRINT 1	Desarrollador 1 Desarrollador 3	✓ Realizar el Análisis y Diseño del proyecto	OK	Done
			✓ Realizar el Modelamiento de la Base de datos		
2	SPRINT 2	Desarrollador 1 Desarrollador 3	✓ Creación de entidades, campos calculados y carga de data en tablas de decisión en InRule	OK	Done
			✓ Creación de la lógica de negocio mediante el lenguaje de reglas en InRule		
3	SPRINT 3	Desarrollador 2 Desarrollador 3	✓ Creación de web service en Visual Studio .Net C#	OK	Done
			✓ Creación de los dos aplicativos: un aplicativo en Visual Studio .Net y otro aplicativo en Visual Basic 6.0		

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

La tabla de la página anterior muestra el estado actual de las tareas asignadas a los responsables, las cuales tienen el estado final de “Hecho” (Done) al cierre del proyecto.

En este capítulo se realizó la Intervención Metodológica, siguiendo lineamientos establecidos en el modelo aplicativo. Se aplicó cada una de las fases, partiendo por la primera denominada Definición del backlog del producto, siguiendo por la Planificación del sprint, después el Scrum diario, continuando con la Revisión del sprint y finalmente realizando la Retrospectiva del sprint. Todo este proceso culmina con la producción de un incremento operativo del software verificado y validado por el cliente, obteniéndose una versión terminada del producto (producto esperado).

CAPÍTULO IV

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En este capítulo se presentarán los resultados y se validará la hipótesis, analizándose los resultados obtenidos después de aplicar la metodología Scrum a un caso práctico real. Se podrá percibir la nueva realidad que se refleja en un mejor clima laboral, el cumplimiento de los tiempos de entrega de los sprints del proyecto en los plazos convenidos y sobre todo la satisfacción de los clientes y colaboradores de CCJ con la nueva metodología aplicada.

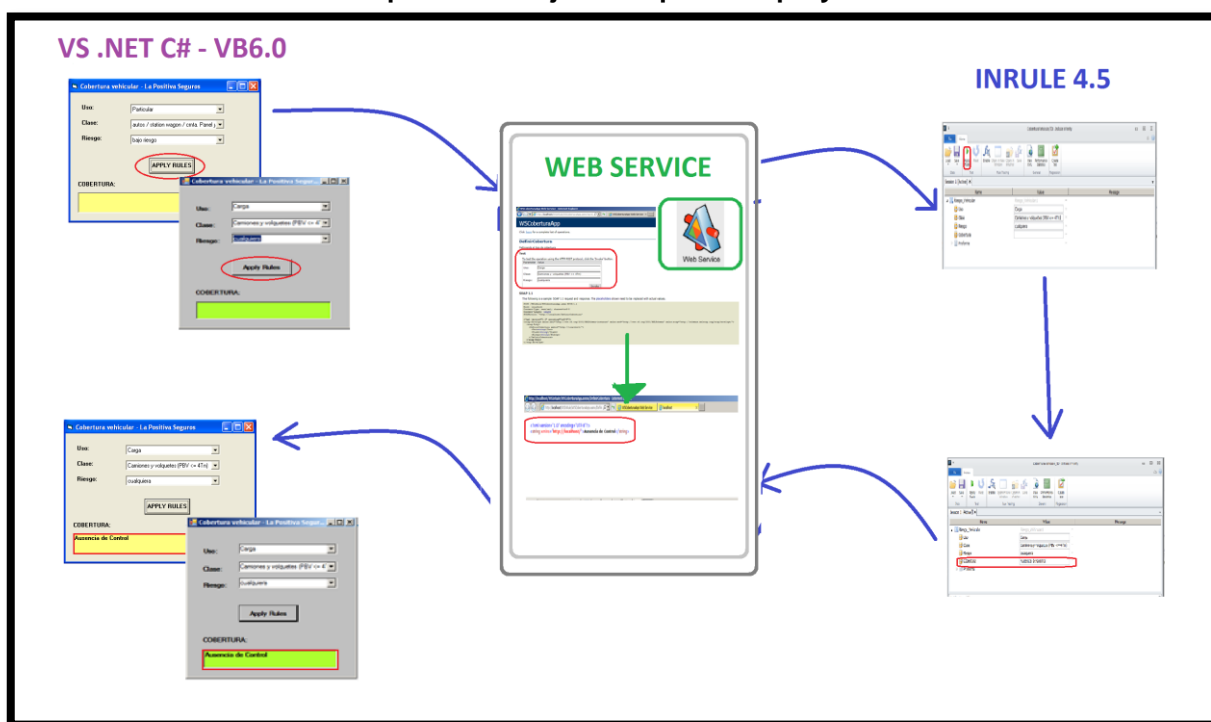
4.1. PRESENTACIÓN DE RESULTADOS

El objetivo inicial del proyecto “Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.”, era proveer al usuario final de un software que le permita ingresar unos parámetros de entrada, aplicar las reglas de negocio (proceso de decisión) y obtener la cobertura vehicular según la lógica del negocio (conjunto de condiciones y acciones).

En el gráfico N° 4.1 se visualiza el esquema que describe el flujo que siguió el proyecto.

Del gráfico de la página siguiente, se observa la secuencia e interacción de los aplicativos construidos, así como las tecnologías involucradas en el proyecto.

Gráfico N° 4. 1
Esquema del flujo descriptivo del proyecto



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

El gráfico N° 4.2 permite visualizar el ingreso de los parámetros de entrada al aplicativo Inrule.

Gráfico N° 4. 2
Parámetros de entrada en Inrule Technology

TarifarioVehicular

Uso	Particular
Oficina	Lima
Producto	Daño Propio
Clase	Automovil
Marca	Alfa Romeo
Modelo	145 1.4 TS 16V
Anio	1998
TimonCambiado	SI

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se observa el ingreso de los parámetros de entrada (data entry) siguiente: Uso, Oficina, Producto, Clase, Marca, Modelo, Anio y TimónCambiado.

El gráfico N° 4.3 permite verificar los parámetros de salida obtenidos después de aplicar la regla de negocio.

Gráfico N° 4. 3
Parámetros de salida en Inrule Technology

TarifarioVehicular:1	
Uso	Particular
Oficina	Lima
Producto	Daño Propio
Clase	Automovil
Marca	Alfa Romeo
Modelo	145 1.4 TS 16V
Año	1998
TimonCambiado	SI
MontoSumaAsegurada	4000
RequiereGPS	No
ObtieneTasa	7,10
PrimaNeta	450

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico se observan los resultados logrados después aplicar la regla de negocio definida con el motor de reglas de negocio Inrule, de acuerdo a las especificaciones proporcionadas por La Positiva Seguros, los mismos que deben verse reflejados en los aplicativos desarrollados y tratados en la Discusión de Resultados.

En el gráfico N° 4.4 se observa la interfaz del aplicativo .Net con el respectivo ingreso de datos.

Gráfico N° 4. 4
Ingreso de parámetros - Software de Cotización de Seguro Vehicular .Net

N° Clase:	Marca:	Modelo:	Año:	Timón cambiado:	Suma asegurada:	Requiere GPS:	Tasa %:	Prima (US\$):
Automovil	Alfa Romeo	145 1.4 TS 16V	1998	SI				

Calcula prima

Prima Neta (US\$):
Derecho de Emisión (US\$):
IGV (US\$):
PRIMA TOTAL (US\$):
PRIMA TOTAL (S/.):

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico de la página anterior se observa el ingreso de los siguientes datos:

Uso: Particular

Oficina: Lima

Producto: Daño Propio

N° Clase: Automóvil

Marca: Alfa Romeo

Modelo 145 1.4 TS 16V

Año: 1998

Timón cambiado: Si

En el gráfico N° 4.5 se observa la interfaz del aplicativo en Visual Basic 6.0 con el respectivo ingreso de datos.

Gráfico N° 4. 5
Ingreso de parámetros - Software de Cotización de Seguro Vehicular VB6.0



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se observa el ingreso de los siguientes datos:

Uso: Particular

Clase: autos / station wagon / cmta. Panel y rural

Riesgo: bajo riesgo

En el gráfico N° 4.6 se observa el ingreso de datos al servicio web “WSCobertura”.

Del gráfico de la página siguiente, se observa el ingreso de los siguientes datos:

Uso: Particular

Clase: autos / station wagon / cmta. Panel y rural

Riesgo: todo riesgo

Gráfico N° 4. 6

Ingreso de parámetros – Servicio Web del Cotizador de Seguro Vehicular

Click [here](#) for a complete list of operations.

Definir Cobertura

Definiendo el tipo de cobertura

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
Uso:	Particular
Clase:	autos / station wagon / cmta. Panel y rural
Riesgo:	todo riesgo

Invoke

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

En la tabla N° 4.1 se observa el resumen de estadísticas de ejecución del motor de reglas de negocio para el presente proyecto.

Tabla N° 4. 1

Estadística de ejecución de Inrule Technology

Summary			
Total execution time:	03.620 seconds	Rule sets fired:	2
State changes:	<u>4</u>	Rules evaluated:	13
Notifications:	<u>0</u>	Total Rule evaluation time:	00.908 seconds
Validation changes:	<u>0</u>	Actions executed:	6
Errors:	<u>0</u>	Total Action execution time:	00.345 seconds

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla se observan diversos indicadores de ejecución, siendo los principales:

Tiempo de ejecución total (Total execution time): 3.620 segundos

Conjunto de reglas lanzadas o ejecutadas (Rule sets fired): 2

Reglas evaluadas (Rules evaluated): 13

Tiempo total de evaluación de reglas (Total Rule evaluation time): 0.908 segundos

Acciones ejecutadas (Actions executed): 6

Tiempo total de ejecución de acciones (Total Action execution time): 0.345 segundos

Estos indicadores de ejecución indican que la aplicación implementada en Inrule Technology: CoberturaVehiculo_TD.ruleapp, ha sido ejecutada con éxito siendo los tiempos de ejecución óptimos (tiempo de ejecución total 3.62 segundos) para una cantidad de data de 10000 registros de la base de datos.

4.2. DISCUSIÓN DE RESULTADOS

La discusión de resultados, estará guiada por la hipótesis general y se debe evaluar los resultados obtenidos después de la aplicación de Scrum al proyecto objeto de medición.

En el gráfico 4.7 se observa la interfaz del aplicativo .Net después de haber aplicado las reglas mediante el botón “Calcula prima”.

Gráfico N° 4. 7

Resultados obtenidos - Software de Cotización de Seguro Vehicular .Net

N° Clase:	Marca:	Modelo:	Año:	Timón cambiado:	Suma asegurada:	Requiere GPS:	Tasa %:	Prima (US\$):
Automovil	Alfa Romeo	145 1.4 TS 16V	1998	SI	4000	No	7.10	450

Prima Neta (US\$):	450
Derecho de Emisión (US\$):	8.52
IGV (US\$):	55.58
PRIMA TOTAL (US\$):	514.10
PRIMA TOTAL (S/.):	514.10

Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

Del gráfico se observa que se obtuvieron los siguientes resultados:

Suma asegurada: 4000

Requiere GPC: No

Tasa %: 7.10

Prima (US\$): 450

Prima Neta (US\$): 450

Derecho de Emisión (US\$): 8.52

IGV (US\$): 55.58

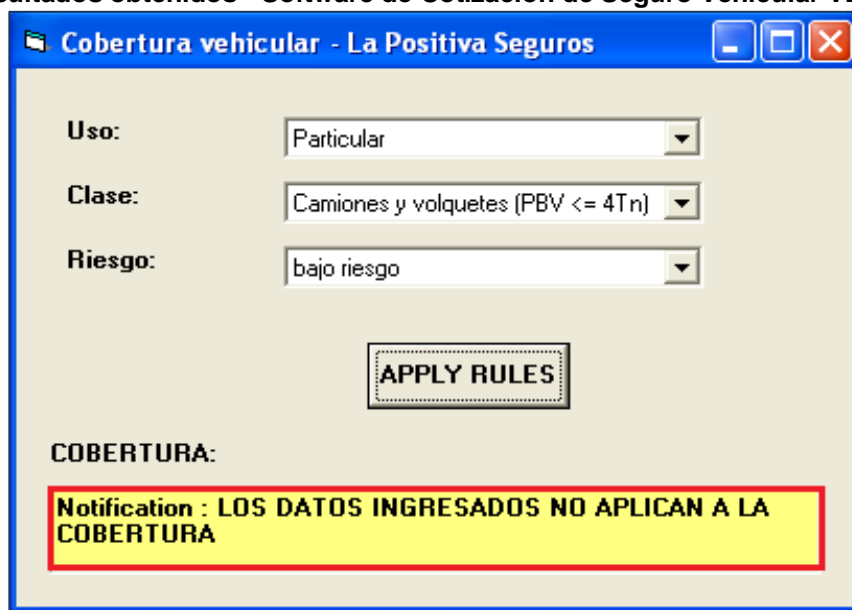
PRIMA TOTAL (US\$): 514.10

PRIMA TOTAL (S/.): 514.10

Los datos arrojados son válidos contrastando con la base de datos proporcionada por el cliente La Positiva Seguros S.A.

En el gráfico 4.8 se observa la interfaz del aplicativo en Visual Basic 6.0 después de haber aplicado las reglas mediante el botón “Apply Rules”.

Gráfico N° 4. 8
Resultados obtenidos - Software de Cotización de Seguro Vehicular VB6.0



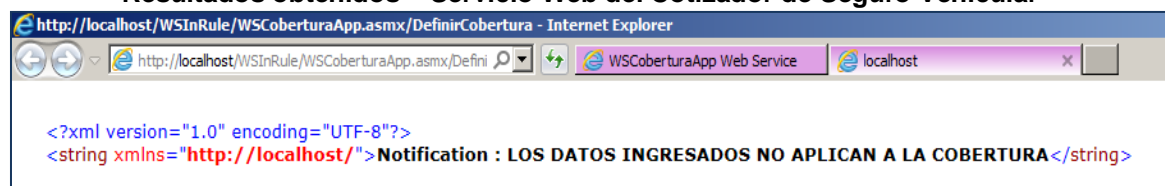
Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico se observa que se obtuvo como resultado para la cobertura, una notificación establecida en la regla de negocios en Inrule que dice: “Notificación: Los datos ingresados no aplican a la cobertura”.

El mensaje de cobertura arrojado es válido dadas las acciones y condiciones en la regla de negocio y su respectivo contraste con la base de datos proporcionada por el cliente La Positiva Seguros S.A.

En el gráfico 4.9 se observa el resultado después de invocar al servicio web.

Gráfico N° 4. 9
Resultados obtenidos – Servicio Web del Cotizador de Seguro Vehicular



Fuente: Unidad de Negocio Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Del gráfico de la página anterior se observa que se obtuvo como resultado para la cobertura, una notificación establecida en la regla de negocios en Inrule que dice: “Notificación: Los datos ingresados no aplican a la cobertura”.

El mensaje lanzado por el servicio web es válido dadas las acciones y condiciones en la regla de negocio y su respectivo contraste con la base de datos proporcionada por el cliente La Positiva Seguros S.A.

4.3. VALIDACIÓN DE HIPÓTESIS

En este punto se busca demostrar la validez o falsedad de la hipótesis general de la tesis referida a: “La Metodología Scrum influye significativamente sobre el incremento de la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C.”.

Para poder validar la hipótesis presentada, se ha considerado establecer la relación entre dos variables que permitirán medir y establecer los indicadores de la hipótesis. Éstas variables se describen de la siguiente manera: “El nivel de conocimiento o grado de dominio de la metodología ágil Scrum por el personal de TI influye sobre su productividad”. La relación es como se muestra a continuación:

$$Productividad = f(\text{Nivel de conocimiento de la Metodología Scrum})$$

Siendo:

Variable Independiente: Nivel de conocimiento de la Metodología Scrum

Variable dependiente: Productividad

La tabla N° 4.2 y la tabla N° 4.3 permite realizar una comparación de los indicadores de productividad, antes y después de aplicada la metodología Scrum respectivamente, aplicando métricas de productividad por cada desarrollador (Datos cuantificados por la Gerencia de TI en base a métricas detalladas en el capítulo I).

Tabla N° 4. 2
Indicadores de productividad ANTES por desarrollador empleado en CCJ S.A.C.

MÉTRICAS DE LA CALIDAD DEL SOFTWARE	DEL PRODUCTO			DEL PROCESO		
	Tamaño (líneas código)	Estructura Datos (cantidad de datos)	Lógica de procedimiento (nivel de conocimiento)	Tiempo de desarrollo (horas - hombre)	Reutilización (código nuevo vs. reusado)	Productividad (salidas / entradas)
Desarrollador 1	0.8	0.8	0.5	0.8	0.7	0.72
Desarrollador 2	0.6	0.7	0.4	0.9	0.6	0.64
Desarrollador 3	0.5	0.6	0.3	0.6	0.4	0.48

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Gerencia de Tecnología de Información

Tabla N° 4. 3
Indicadores de productividad DESPUÉS por desarrollador empleado en CCJ S.A.C.

MÉTRICAS DE LA CALIDAD DEL SOFTWARE	DEL PRODUCTO			DEL PROCESO		
	Tamaño (líneas código)	Estructura Datos (cantidad de datos)	Lógica de procedimiento (nivel de conocimiento)	Tiempo de desarrollo (horas - hombre)	Reutilización (código nuevo vs. reusado)	Productividad (salidas / entradas)
Desarrollador 1	1.0	1.0	0.9	1.0	0.9	0.96
Desarrollador 2	0.9	1.0	0.9	0.9	1.0	0.94
Desarrollador 3	0.9	0.9	0.8	0.9	0.9	0.88

Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.

Elaboración: Gerencia de Tecnología de Información

De las tablas anteriores, se puede evidenciar el incremento de productividad por cada desarrollador antes de aplicar Scrum y después de aplicar Scrum. El Desarrollador 1 presenta un incremento de productividad de 0.72 a 0.96 en promedio, el Desarrollador 2 presenta un incremento de productividad de 0.64 a 0.94 en promedio, y el Desarrollado 3 presenta un incremento de productividad de 0.48 a 0.88 en promedio.

La tabla N° 4.4 muestra los tiempos asignados versus los tiempos realmente utilizados por cada desarrollador en el proyecto en estudio.

Tabla N° 4. 4
Cuadro de Tiempo Asignado vs. Tiempo Utilizado por Desarrollador

Ítem	Nombre del Proyecto	Número de Desarrollador	Tiempo Asignado (días)	Tiempo Utilizado (días)	Tiempo perdido (días)	Tiempo perdido (horas)
001	Implementación de las reglas de negocio para establecer la cobertura vehicular en La Positiva Seguros S.A.	Desarrollador 1	10	10	0	0
		Desarrollador 2	20	20	0	0
		Desarrollador 3	10	10	0	0

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

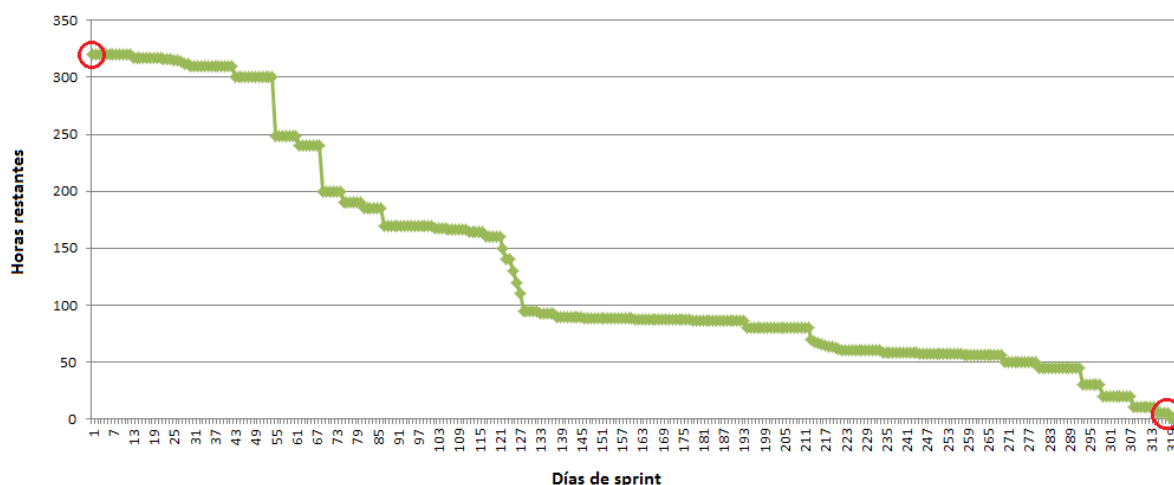
Elaboración: Propia

La tabla muestra que cada desarrollador culminó las tareas asignadas en cada sprint en el tiempo asignado, es decir el tiempo asignado es igual al tiempo utilizado. Siendo el valor del tiempo perdido de 0 días. Por lo tanto, no se incurrieron en costos adicionales en la ejecución del proyecto.

El gráfico N° 4.10 muestra la tendencia del proceso de desarrollo ágil Scrum para el proyecto en estudio de principio a fin.

Del gráfico lineal de la página siguiente, acerca del proyecto aplicando Scrum, muestra la curva del avance midiendo el tiempo restante (puntos de historia en horas) en el eje y, y las fechas de entrega en días en el eje x.

Gráfico N° 4. 10
Gráfico lineal de la tendencia del proceso de desarrollo ágil Scrum – Proyecto Cotizador Vehicular



Fuente: Unidad de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

Se observa que al inicio del proyecto, el Día 1 el avance era de 0% es decir las tareas del product backlog están a tope en un 100% (40 días = 320 horas). A medida que transcurre el proceso ágil se tiene que en el Día 10 se logró un avance y las tareas se redujeron a un 80% aproximadamente. Finalmente al cierre del proyecto, el Día 40 se logró que las tareas se redujeran a un 0%.

La tabla N° 4.5 permite percibir la nueva realidad que se avizoraba con la aplicación de la metodología Scrum, la cual se refleja en la satisfacción de los trabajadores involucrados en el desarrollo de los proyectos de software en CCJ.

Tabla N° 4. 5
Cuadro que refleja la satisfacción de los trabajadores aplicando Scrum

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	En desacuerdo
¿Está de acuerdo con la metodología de desarrollo de software usada actualmente?	70%	20%	10%	-
¿Está de acuerdo que ésta metodología permite optimizar tiempos y costos?	65%	30%	5%	-
¿Está de acuerdo en que ésta metodología permite mejorar las líneas de comunicación?	80%	20%	-	-
¿Está de acuerdo en que ésta metodología permite la existencia de un adecuado clima laboral?	80%	15%	5%	-
¿Estaría de acuerdo con la propuesta de trabajar con la metodología Scrum para el desarrollo de software?	90%	10%	-	-

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.
Elaboración: Propia

De la tabla de la página anterior se puede notar que existe conformidad y un acuerdo generalizado con la metodología empleada actualmente, la cual permite optimizar tiempos y costos, también genera un manejo adecuado de las líneas de comunicación y crea un mejor clima laboral. Se resalta que en el ítem 5 un 90% está de acuerdo con trabajar con Scrum como metodología de cabecera.

La tabla N° 4.6 permite percibir la nueva situación de satisfacción de los clientes finales quienes hacen uso de los servicios de tecnologías de información por parte de CCJ.

Tabla N° 4. 6
Cuadro que refleja la satisfacción de los clientes aplicando Scrum

Ítem	Muy de acuerdo	De acuerdo	Poco de acuerdo	Muy en desacuerdo
¿Está de acuerdo con los plazos establecidos para su proyecto?	20%	65%	15%	-
¿Está de acuerdo con los costos establecidos para su proyecto?	-	60%	40%	-
¿Cree Ud. que la empresa cuenta con tecnologías de vanguardia?	-	60%	30%	10%
¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto?	-	60%	30%	10%
¿Está de acuerdo en que la empresa termina su proyecto en los plazos y tiempos establecidos?	65%	15%	20%	-
¿Está de acuerdo que cuando la empresa culmina su proyecto Ud. sale conforme?	55%	25%	15%	5%

Fuente: Unidad de Negocio de Tecnologías de Información – CCJ S.A.C.

Elaboración: Propia

De la tabla se puede notar que existe conformidad y un acuerdo por parte de los clientes o usuarios finales con los plazos y costos establecidos para la realización de un proyecto, además un 55% está muy de acuerdo ante la pregunta ¿Está de acuerdo que cuando la empresa culmina su proyecto Ud. sale conforme? Éste indicador es bueno considerando que inicialmente solo el 10% estaba muy de acuerdo con el mencionado ítem.

Todos estos resultados cuantitativos obtenidos, permiten validar la hipótesis lanzada al inicio de la presente tesis como VERDADERA. Por lo que, se sustenta la siguiente afirmación: “Si el personal de TI domina la metodología ágil Scrum y lo emplea en cada proceso o trabajo diario de manera correcta, entonces la influencia sobre su productividad en el proceso de desarrollo de software será significativamente positiva”.

En este capítulo se pudo conocer el Análisis de Resultados, que involucra la nueva realidad, los nuevos comportamientos de las variables en términos de productividad, eficiencia, menores costos y tiempos de ejecución de los proyectos. Es decir la nueva realidad supera a la realidad inicial ($R2 \gg R1$). Afirmándose de ésta manera la hipótesis planteada en el capítulo I. Se puede deducir que la Metodología Scrum sí influye significativamente sobre el incremento de la productividad del proceso de desarrollo de software en la Empresa CCJ S.A.C.

CONCLUSIONES

1. Después de aplicar Scrum al proyecto se concluye que, la Metodología Scrum influye positivamente sobre el incremento de la productividad del proceso de desarrollo de software en términos de reducción de tiempos y costos, logrando que los proyectos se realicen en los plazos estimados en la propuesta técnica, existiendo un desfase de cero días y una pérdida neta de S/. 0.00 nuevos soles.
2. El proceso de desarrollo iterativo y creciente que propone el Modelo Aplicativo Scrum permite concluir que, ésta metodología ágil hace factible el planificar, ordenar, reportar el trabajo del día a día, semanal, mensual y anual, impulsando la creación de equipos auto-organizados integrando a todos los miembros del equipo y disciplinas involucradas en el proyecto, creando un mejor clima laboral.
3. Se conoce que cualquier desarrollo de software parte de un mismo problema: conocer las necesidades de los clientes, Scrum da mayor detalle a las partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo normalmente de 1 a 4 semanas. Cada uno de estos periodos de desarrollo concluye con la producción de un incremento operativo del producto, obteniendo una correcta estimación del tiempo presentado en la propuesta técnica al cliente logrando así una mayor credibilidad y confianza de éste sobre la empresa CCJ.
4. La aplicación de la tesis permitió cristalizar una nueva realidad que es muy superior a la realidad inicial ($R2 \gg R1$), ello en términos de nuevos comportamientos de las variables: La productividad se incrementó en un 30%, los costos adicionales por mala estimación se redujeron a S/. 0.00 nuevos soles y el tiempo de ejecución del proyecto se cumplió dentro de la fecha no existiendo desfase alguno.

RECOMENDACIONES

1. Después de aplicar Scrum al proyecto se recomienda que, ésta metodología ágil debe ser considerada una variable importante ya que quedó demostrado que influye de manera positiva sobre el incremento de la productividad del proceso de desarrollo de software, sobre todo para organizaciones que requieren rapidez de desarrollo, agilidad en la migración de una tecnología a otra y poseen alta rotación de personal.
2. La Metodología Ágil Scrum al ser un proceso de desarrollo iterativo y creciente, se recomienda su empleo para cualquier área de una empresa como TI, Operaciones, Administración, entre otras, porque permite la creación de equipos auto-organizados impulsando la comunicación verbal entre todos los integrantes y disciplinas involucradas en un proyecto, como en la presente tesis que involucra a personal de distintas carreras profesionales.
3. La aplicación del modelo aplicativo Scrum de manera correcta fase por fase permite obtener los resultados esperados por parte del dueño del producto, pudiendo encontrarse errores al final de cada sprint que pueden corregirse en la fase de retrospectiva del sprint, para finalmente obtener el producto esperado por el cliente.
4. La continuidad del incremento de la productividad está asegurada siempre y cuando los objetivos de los colaboradores sigan alineados con los de la empresa, porque una metodología ágil está más enfocada a las personas que a los procesos, y se recomienda elaborar un plan de mantenimiento para permitir la continuidad del producto frente a los continuos cambios del mercado empresarial.

REFERENCIAS

REFERENCIAS BIBLIOGRÁFICAS

1. Hernández Sampieri, R. & Fernández Collado, C. & Baptista Lucio, P. (1991). Metodología de la Investigación. Estado de México: Editorial McGraw Hill Interamericana de México.
2. Kniberg, H. (2007). Scrum y XP desde las Trincheras: Cómo hacemos Scrum. Estados Unidos: Editorial C4Media.
3. Mudarra Teruel, H. & Pons Aróztegui, J. (2010). Automatización de Sistemas de Desarrollo Ágil – Scrum: Team & Role. Memoria del Proyecto de Fin de Carrera de Ingeniería Informática. Barcelona, España.
4. Nakashima Chávez, G. J. (2009). Mejora del Proceso de Software de una Empresa Desarrolladora de Software: Caso Competisoft – Perú Delta. Tesis para Optar por el Título de Ingeniero Informático. Lima.
5. Palacio, J. & Ruata, C. (2011). Scrum Manager Gestión de Proyectos. Feria Informática. Barcelona, España.
6. Rodríguez, G. P, (2008). Estudio de la Aplicación de Metodologías Ágiles para la evolución de productos software. Tesis de Máster en Tecnologías de la Información, Facultad de Informática. Madrid.

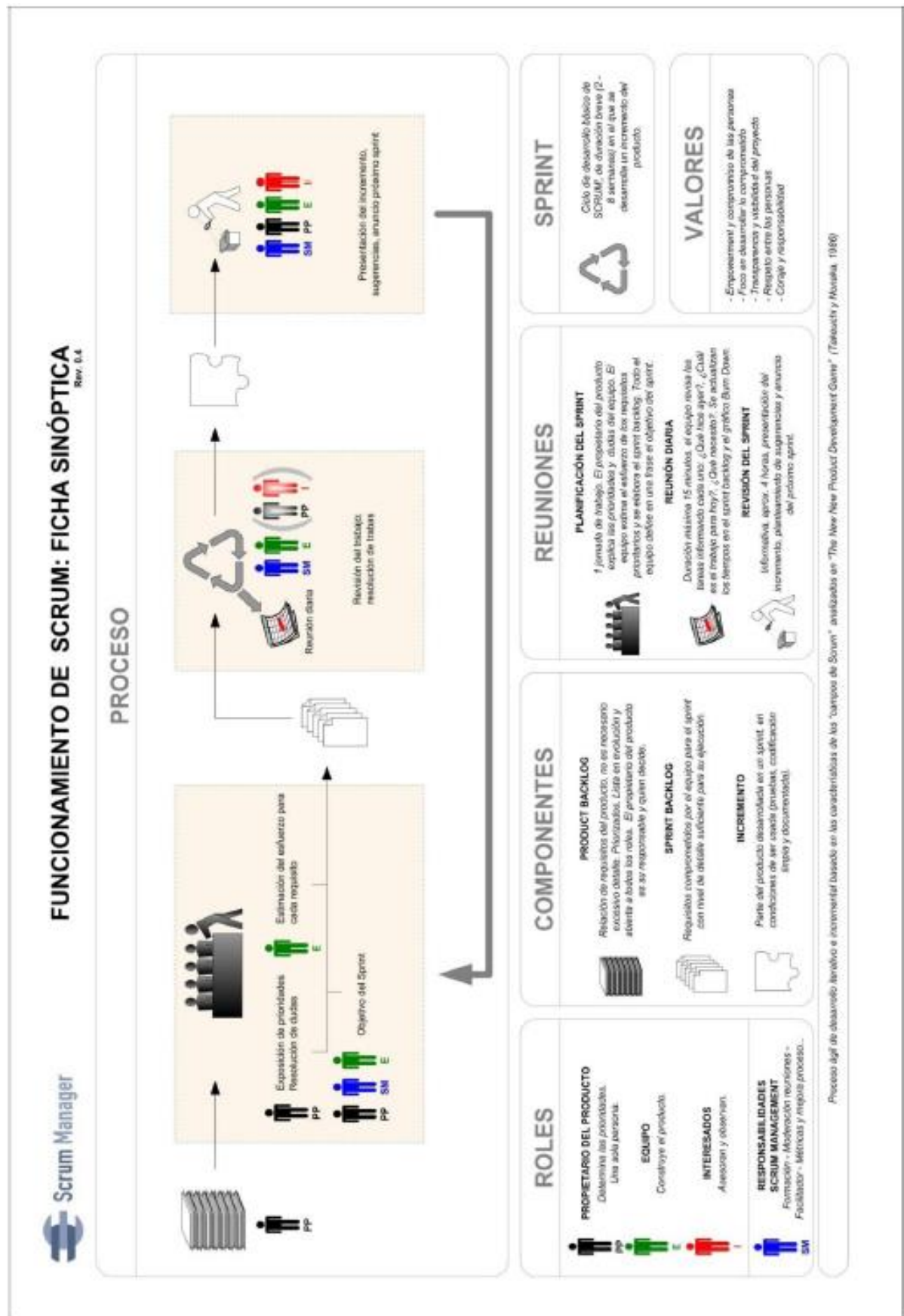
REFERENCIAS ELECTRONICAS

1. Gestión de productividad de desarrollo. ¿Cuánto vale el kilo de software?.
Disponble en: "<http://www.dosideas.com/noticias/metodologias/981-scrum-en-1-sola-pagina.html>"
Accesado el: [14 diciembre 2013]
2. Medida de la productividad del proceso de desarrollo del software.
Disponble en: "<http://www.buenastareas.com/ensayos/Medida-De-La-Productividad-Del-Proceso/2455315.html>"
Accesado el: [03 marzo 2014]

3. Palacio Juan; Ruata Claudia (2011). Scrum Manager Gestión de proyectos.
Disponibile en: "http://www.scrummanager.net/files/sm_proyecto.pdf"
Accesado el: [05 febrero 2014]
4. Palacio Juan (2006). El modelo Scrum.
Disponibile en: "http://www.navegapolis.net/files/s/NST-010_01.pdf"
Accesado el: [10 febrero 2014]
5. Scrum Alliance Core Scrum v2012.12.13. Todo Scrum en una sola página.
Disponibile en: "<http://www.dosideas.com/noticias/metodologias/981-scrum-en-1-sola-pagina.html>"
Accesado el: [20 febrero 2014]
6. Telaya Luis (2012). Implementación de una red social usando metodologías ágiles para el proceso de participación estudiantil en la Universidad Autónoma del Perú.
Disponibile en: "<http://dc364.4shared.com/doc/T0kNVxLn/preview.html>"
Accesado el: [07 marzo 2014]

ANEXOS

ANEXO I. Visión General del Modelo Scrum



ANEXO II. Encuesta aplicada sobre la percepción de la Metodología de Desarrollo de Software actualmente empleada en la unidad de negocio de TI - CCJ S.A.C.



ENCUESTA DE APRECIACIÓN DEL USO DE LA METODOLOGÍA DE DESARROLLO DE SOFTWARE ACTUAL EMPLEADA EN CCJ S.A.C.

Marque con un aspa (X) la alternativa que considere más adecuada.

Fecha: / /

Item	Muy de acuerdo	De acuerdo	Poco de acuerdo	En desacuerdo
¿Está de acuerdo con la metodología de desarrollo de software usada actualmente?				
¿Está de acuerdo en que esta metodología permite optimizar tiempos y costos?				
¿Está de acuerdo en que esta metodología permite mejorar las líneas de comunicación?				
¿Está de acuerdo en que esta metodología permite la existencia de un adecuado clima laboral?				
¿Estaría de acuerdo con la propuesta de implementar una metodología ágil para el desarrollo de software?				

Item	Muy de acuerdo	De acuerdo	Poco de acuerdo	Muy en desacuerdo
¿Está de acuerdo con los plazos establecidos para su proyecto?				
¿Está de acuerdo con los costos establecidos para su proyecto?				
¿Cree Ud. que la empresa cuenta con tecnologías de vanguardia?				
¿Cree Ud. que la empresa cuenta con diversidad de tecnologías para poner en marcha su proyecto?				
¿Está de acuerdo en que la empresa termina su proyecto en los plazos y tiempos establecidos?				
¿Está de acuerdo que cuando la empresa culmina su proyecto Ud. sale conforme?				

¿Usted está de acuerdo en cambiar la Metodología de Desarrollo de Software que se aplica actualmente a los proyectos de la Unidad de negocio de TI - CCJ S.A.C.? Si su respuesta es NO, enumere tres razones del por qué.

.....

.....

.....

.....

La presente encuesta es de carácter confidencial, anónima y con fines de investigación.

ANEXO III. Documento excel conteniendo data de las reglas de negocio proporcionado por el cliente La Positiva Seguros S.A.

[illegible]

ANEXO IV. Fotografías de los talleres prácticos de la aplicación de Scrum al caso de estudio



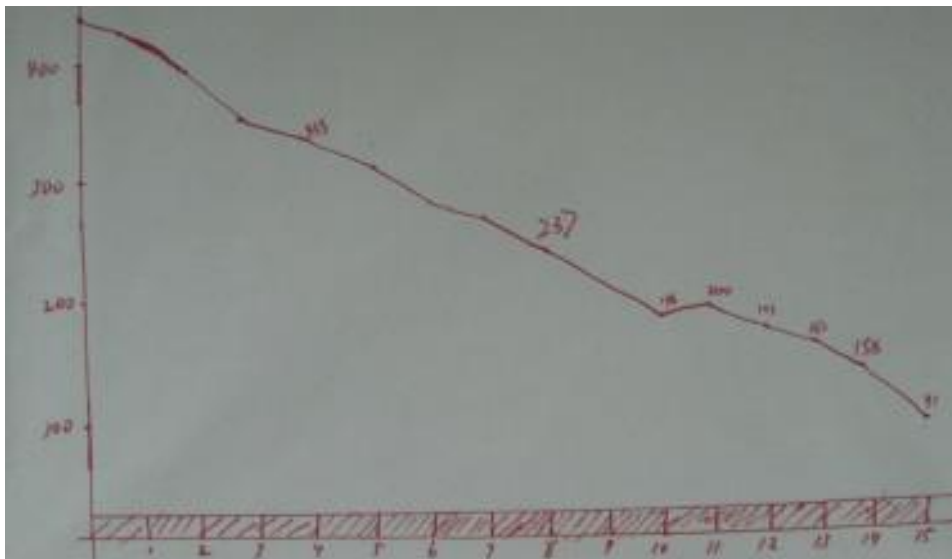
Fotografía N° 01: Juego de roles – Proyecto Inrule Technology



Fotografía N° 02: Sprint Backlog – Proyecto Inrule Technology



Fotografía N° 03: Personal de TI participando en las reuniones



Fotografía N° 04: Diagrama de BurnDown del Sprint