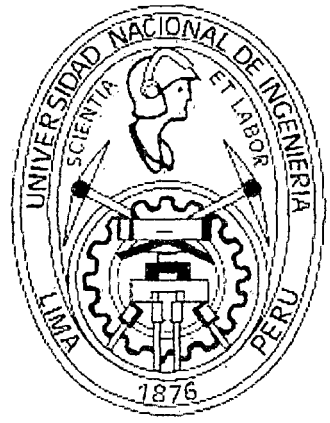
UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

SECCIÓN DE POSGRADO



"METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE SOFTWARE PARA UNA FÁBRICA DE SOFTWARE"

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS CON MENCIÓN EN:

INGENIERÍA DE SISTEMAS

ING. ROBERT PERCY OCOLA AGÜERO

LIMA – PERÚ

2012

DEDICATORIA

Dedico esta tesis a Dios por darme sabiduría, fortaleza y guiarme por el camino de la luz a través de su infinita bondad.

A mis padres Percy y Vilma por el soporte idóneo y por el amor brindado para la culminación de este trabajo.

AGRADECIMIENTO

A mis Profesores, amigos y colegas que aportaron con un grano de arena en la culminación de la presente tesis.

A mis hermanos Christian y Kendy por su comprensión y apoyo para lograr superarme en mi formación profesional.

¡Que Dios les bendiga!

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

DESCRIPTORES TEMÁTICOS

- Metodología de Desarrollo
- Metodología Ágil
- Fábrica del Software
- Ingeniería del Software
- Proyectos Ágiles
- Evolución del Software
- Modelo de Ciclo de vida
- Proyecto Adaptativo y Prescriptivo
- Desarrollo de Software
- Mantenimiento de Software

RESUMEN

Actualmente la fábrica de software se encarga de ejecutar proyectos de desarrollo y mantenimiento de sistemas de información para empresas y entidades estatales. Esta tesis plantea proponer una nueva metodología que mejore el desarrollo y mantenimiento de la fábrica de software. Para ello se realizó un análisis de las diversas fases de desarrollo de la metodología actualmente existente y con los resultados obtenidos se propuso una metodología que agregue valor. Finalmente se realizó la aplicación sobre un requerimiento de software a fin de demostrar que el uso de la nueva metodología cumple con lo propuesto.

ABSTRACT

Currently the software factory is responsible for implementing development projects and maintenance of information systems for companies and government agencies. This thesis proposes a new methodology to improve the development and maintenance of the software factory. In order to commit our goal, an analysis was carried out in the various stages of development currently existing methodology and results proposed a methodology that adds value. Finally, based on an application software requirement we demonstrate that the use of the new methodology complies with the objective.

INTRODUCCIÓN

La fábrica de software está orientada a brindar servicios especializados en tecnologías innovadoras que aporten valor agregado a los procesos de negocios de sus clientes. Así, su concepto de solución de tecnologías de la información, como respuesta a una necesidad de cambio y optimización, se desarrolla sobre los componentes de conocimiento de los procesos de negocio; especialización de los recursos humanos y selección de la tecnología que facilita el cambio.

En este contexto, se plantea como problema la mejora de la actual metodología de desarrollo y mantenimiento de software considerando los aportes de metodologías tradicionales y ágiles. En ese sentido, la hipótesis afirma la factibilidad de diseñar una nueva metodología de desarrollo y mantenimiento de software a fin que mejore la atención de requerimientos de los sistemas de información. Es decir, se demuestra que es posible identificar las mejoras prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software; se puede enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología; se puede diseñar la metodología mediante la combinación de las metodologías de desarrollo como el Proceso Unificado de Desarrollo [RUP], y de metodologías ágiles [como el XP[1] y Scrum[2]]; y se puede aplicar la nueva metodología de desarrollo en la construcción de un producto de software.

---

[1] Programación Extrema o XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Para una mejor compresión de los resultados de las hipótesis, la presente tesis está estructurada en seis capítulos, las conclusiones y recomendaciones, el soporte bibliográfico y anexos. En el Capítulo I se presenta el problema de la investigación de la tesis. El Capítulo II ofrece una visión detallada del marco teórico de las metodologías de desarrollo. Se estudian las relaciones existentes entre las distintas orientaciones. Se identifican carencias en las propuestas presentadas hasta el momento, y se proponen líneas de investigación abordadas en el resto del trabajo. El Capítulo III menciona la metodología de investigación de la ingeniería del software. El Capítulo IV presenta el nuevo método de desarrollo. Para ello, identifica los aspectos en común de metodologías de desarrollo existentes. El Capítulo V presenta la ejecución de la metodología de desarrollo y mantenimiento a un cliente [entidad pública] de la fábrica de software. Finalmente, el Capítulo VI presenta el análisis de los resultados, el cual permite justificar la aceptación de las hipótesis planteadas.

Adicionalmente a esta estructura, se presentan una serie de anexos que han sido separados de la disertación principal con objeto de mejorar la comprensión de la exposición, y cuyo contenido se detalla a continuación: Definición de mediciones, diagrama del proceso de mantenimiento de la fábrica de software, resultados de retrospectiva y términos y definiciones.

---

[2] Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

CAPÍTULO I

PROBLEMA DE INVESTIGACIÓN

1.1 DEFINICIÓN DEL PROBLEMA

Según [Guibovich, 2011], los principales problemas que actualmente afectan a la fábrica de software están relacionados con la insatisfacción del cliente, el incumplimiento de los acuerdos y niveles de servicio, y producto con errores.

Respecto a la insatisfacción del cliente, identifica como factores explicativos los relacionados al medio ambiente, factor tiempo, personal, producto y procedimientos.

Esta insatisfacción se deriva a su vez del incumplimiento de los acuerdos y niveles de servicio [ANS]. Los cuales se explican por los desfases de la parte de requerimientos y cambios frecuentes de alcance de las necesidades de cliente [por ejemplo, modificación de los tiempos]; conduciendo en varias ocasiones en la programación de actividades para ser concluidos en el siguiente ciclo.

El incumplimiento de los acuerdos y niveles de servicio incide, a su vez, en los productos con errores. Al respecto, se menciona que no se cuenta con herramientas que permitan identificar errores antes de que el producto sea entregado al cliente. En la siguiente tabla se brinda mayores detalles de los factores del problema.

Insatisfacción del cliente	Incumplimiento de los acuerdos y niveles de servicio	Producto con errores
En el medio ambiente Originadas por el cliente. • Mala planificación. La fecha límite a nivel organizacional no se prevé con anticipación. • Políticas. Para el caso de entidades del Estado, las políticas públicas de gobierno o normas que pueden entrar en vigencia. • Metodología en la solicitud de requerimientos. • Requerimientos mal definidos. **Rotaciones de personal** • Renuncias inesperadas del personal. • Cambios internos entre áreas de desarrollo y mantenimiento. **Políticas de la organización.** • Clima organizacional. • Integración entre áreas. **El factor tiempo** • Entregas fuera de plazo. • Mala planificación del análisis de tiempo para la atención del producto o servicio. • Reprogramaciones. • Indisponibilidad del usuario. **El personal que participa del proceso de software**	**Desfases de los Requerimientos.** • Desfases en los tiempos. • Por cambios de alcance mal gestionados sin usar el procedimiento vigente de la fábrica de software. • Por dependencias con el personal del cliente. • Por dependencias funcional o técnica con otros requerimientos. • Por configuración de ambiente. • Postergaciones de pruebas funcionales y de sistemas, las cuales no permiten llegar a la fecha acordada. • Ingresos de Requerimientos urgentes. • Postergaciones en los pases al ambiente de pruebas QA[3] o de producción. • Por priorización de otros Requerimientos (Nuevos Requerimientos o cambios de prioridades) • Por errores. • Por errores encontrados al momento de las pruebas.	**Alcance del Requerimientos.** • Falta conocer el alcance del Requerimientos. • Relación con otros sistemas. • Implementar validaciones funcionales. • Cuando se implementa nuevas funcionales y no se verifica su impacto total en el sistema. • Verificación de reportes finales a solicitud del usuario. • Funcionalidad del sistema. **Ambiente y configuración.** • Olvido de tareas asignadas. • No se haga uso de los formatos estándares de la fábrica de software. • Configuración no adecuada de los ambientes de pruebas hace que muchas veces no se llegue a trasmitir todos los objetos necesarios del aplicativo, que no se verifique detalladamente los requisitos para el pase a calidad o producción y por lo tanto el producto no se pruebe adecuadamente. • Generación de

[3] QA (Quality Assurance), es un término técnico que se usa en la fábrica de software para hacer referencia a la simulación del pase a producción de un proceso de software a un ambiente similar al de producción tanto a nivel de base de datos como a nivel del servidor de aplicaciones.

Insatisfacción del cliente	Incumplimiento de los acuerdos y niveles de servicio	Producto con errores
• Falta de conocimiento técnico. • Falta de cultura de innovación. • Falta de cultura de investigación la cual puede crecer más por la falta de incentivos por parte de la fábrica de software. • Falta de herramientas para la investigación. • Falta de experiencia. • Proceso de evaluación inmaduro. • Remuneración no acorde con el mercado. • Personal con poca experiencia. **El producto o servicio.** • Producto con error. • Falta de herramientas para ejecutar pruebas de calidad. • Mal entendimiento del requerimiento. • Falta de participación del cliente. • Falta de personal con conocimiento técnico. • Dependencias técnicas. • Pruebas de validación incompletas. • Requerimiento mal especificado. • Documentos mal especificados o definidos hacen que los procedimientos de atención no sean los adecuados. • Sobreesfuerzo. • Cambio de alcance. **Los procedimientos** • Metodología. • Procesos no integrados. • Procesos mal	• Por errores encontrados en los documentos generados por la fábrica de software. **Cambios de alcance.** Ampliación de tiempos. Requerimientos mal definidos. El cliente se da cuenta que no existe alguna funcionalidad que no ha sido contemplada en el transcurso de la atención del requerimiento. Estos cambios de alcance por la premura del tiempo no llegan a ser gestionados según procedimiento y el cliente se ve obligado ante su inconformidad a aprobarlos, obteniendo al final insatisfacción del cliente del producto o servicio final. Ampliación de alcance durante la etapa de elaboración. Ampliación del tiempo por cambio de alcance no aprobado. Cambios de alcance aprobados por el cliente. Aprobaciones automáticas. Documento de análisis aprobado automáticamente. Cambios de alcance al momento de las pruebas. **Paralización de los Requerimientos.** Indisponibilidad del	códigos para insertar funcionalidad. • Carga de objetos en sistema de configuración de versiones. • Requisitos para el pase a producción. • Declaración de parámetros • Registros de datos del Requerimientos. • Auditoría de tablas • Alineación de la base de datos. • Alineación de la base de datos de ambiente de pruebas con el de desarrollo. • Configuración de usuarios en base de datos. **Lenguajes de Programación.** • Falta de experiencia. Se ha validado que la generación de código de programación errónea es la causa que más se repite en la construcción de un producto, que al ser detectados toma mayor tiempo su corrección. • Falta de verificación del código digitado y la mala definición de objetos de base de datos. **Estándares y Documentación.** • Desconocimiento de los estándares. La falta de conocimiento de los estándares en la elaboración de los documentos, en el diseño de la base de

Insatisfacción del cliente	Incumplimiento de los acuerdos y niveles de servicio	Producto con errores
definidos. • Proceso no estandarizados. • Accesos limitados a la información. • Falta de cultura en procesos. • Tecnología. • Falta de recursos de hardware. • Falta de ambiente de pruebas. • Herramientas no automatizadas. • Falta de herramientas integradas (software) adecuadas para la implementación. • Herramientas no estandarizadas.	cliente. Paralización de pruebas funcionales o de sistemas por indisponibilidad del usuario. Paralización en pase a producción por el centro de cómputo del cliente. Indisponibilidad del personal del cliente (vacaciones, capacitaciones, operativos, etc.) Ingresos de nuevos Requerimientos. Por priorización de otros Requerimientos que por la urgencia de estos, hacen que la atención planificada varíe llegando a obtener desfases en el tiempo de entrega de los productos o servicios. **Programado para concluir en otro ciclo.** Requerimientos en proceso. Requerimientos que inician en las últimas semanas del ciclo y que por su complejidad puede ampliarse su atención hasta el siguiente ciclo. Ingreso de Requerimientos por disponibilidad de línea de producción. Requerimientos cuyo desarrollo tomó más de un ciclo de producción. Requerimientos que al finalizar en ciclo se encuentra en estado de espera por lo tanto	datos y en la programación usados en la fábrica de software, este desconocimiento suele plasmarse en la inadecuada actualización de las versiones de los documentos de manejo de estándares. • Alineación de fechas de objetos compilados y las fuentes. • Estándares de programación. • Error en redacción de comentarios. • Nomenclatura. • Error de escritura en la extensión de objetos. • Actualización de las versiones correctas de las fuentes y objetos. • Estándares de documentación. • Estándares de base de datos. • Subir documentos al sistema de control de versiones en la ubicación incorrecta. • Documentos mal detallados e incompletos. Debido a la falta de actualización en el documento de análisis, los procesos de ejecución de pase al ambiente de pruebas o producción no ordenados. • Falta implementar casuística de pruebas. • Falta actualización

Insatisfacción del cliente	Incumplimiento de los acuerdos y niveles de servicio	Producto con errores
	no llega a ser contabilizado para el total de requerimientos atendidos en el presente ciclo de producción. Requerimientos que concluyeron en el ciclo vigente pero no se actualizó su estado en el Sistema de Atención de Requerimientos del cliente.	de documento de análisis. • Procesos de ejecución de pase no ordenados.

Tabla 1: Principales problemas que afectan a la fábrica de software. Fuente: [Guibovich, 2011]

Asimismo, [Guibovich, 2011] evalúa los entregables con defectos con datos estadísticos. Es decir, evalúa las iteraciones del producto, las cuales permite conocer a mayor detalle los problemas que actualmente afectan a los procesos en la fábrica de software. En efecto, concluye que las actividades de Implementación de la Solución, Pruebas Unitarias, y Pruebas Funcionales y de Sistemas generan en promedio un mayor número de errores. A nivel de sub actividades, resalta que se debe poner atención en la elaboración de casos de prueba, lo cual podría estar explicando el retraso de las actividades que se encuentran interrelacionadas con ellas. A continuación se brinda el soporte estadístico.

Ciclo[4]	Total Iteraciones por Ciclo	Requerimiento Defectuosos por Ciclo	Req. Terminados por Ciclo
12	16	10	40
13	11	11	42
14	14	11	55
15	18	11	55
16	32	26	67
17	21	13	70
TOTAL	**112**	**82**	**329**

Fuente: [Guibovich, 2011]

Según la tabla, en el ciclo de producción 16 se generaron el mayor número de iteraciones, y por ende mayores requerimientos defectuosos. Luego evaluó los subprocesos para la etapa de Ingeniería, Subproceso de Construcción del ciclo 16, teniendo el número máximo de iteraciones como sigue a continuación.

SUBPROCESO	ITERACION
Implementación de la Solución	8
Pruebas Unitarias	7
Pruebas Internas	4
Control de la Calidad	4
Pruebas Funcionales y de Sistemas	8
Total General	**31**

Fuente: [Guibovich, 2011]

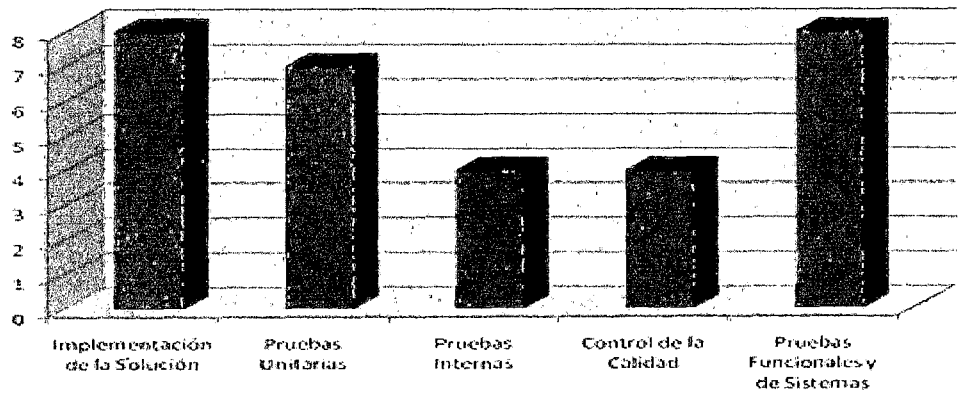


Figura 1.1: Número de iteraciones por subproceso [Guibovich, 2011].

[4] Se refiere al rango de tiempo que se toma como referencia para realizar el informe mensual del servicio realizado por la fábrica de software.

De esta manera, se concluye que las actividades de Implementación de la Solución, Pruebas Unitarias, y Pruebas Funcionales y de Sistemas generan en promedio un mayor número de errores.

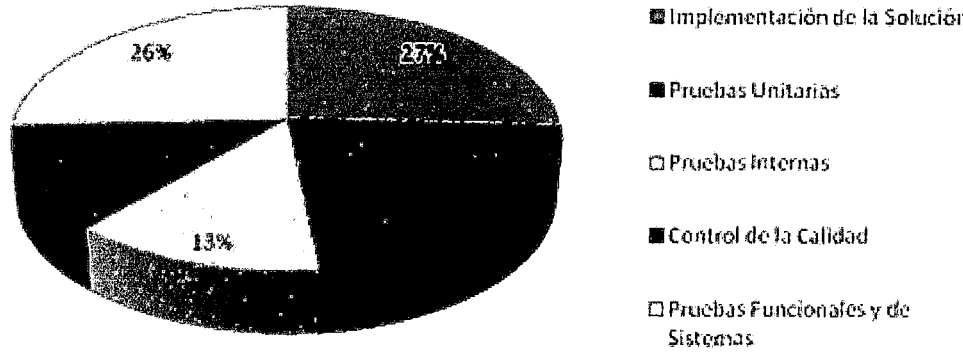


Figura 1.2: Porcentaje de iteraciones por subproceso. [Guibovich, 2011]

A continuación se analiza cada una de las actividades [detalladas en el grafico superior] que tienen mayor porcentaje de error.

- **ACTIVIDAD: Implementación de la Solución [Construcción]**

Actividades	Iteraciones
Implementación de la Solución	8

- **ACTIVIDAD: Pruebas Unitarias.**

Sub-Actividades	Iteraciones
Elaboración de Casos de Prueba	7
Actualización del Sistema de Control de Versiones	3
Envío de los casos de Pruebas	3
Llenar checklist [Analista y Programador]	3
Pruebas Unitarias	3

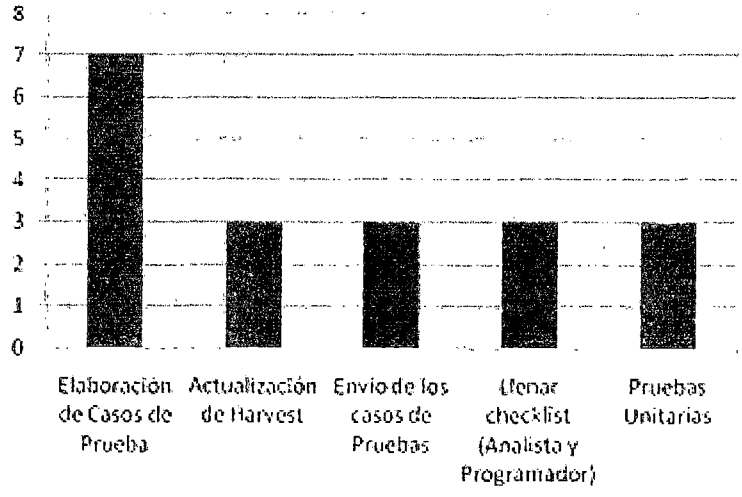


Figura 1.3: Número de iteraciones por subproceso Pruebas unitarias. [Guibovich, 2011]

Del gráfico se puede observar que la sub actividad de Elaboración de Casos de Pruebas tiene un mayor número de errores para el subproceso de Pruebas Unitarias.

- **ACTIVIDAD: Pruebas Funcionales y de Sistemas**

Sub-Actividades	Iteraciones
Pruebas funcionales y aceptación	8
Actualización de Documento de Pruebas Funcionales	3
Aprobación Pruebas Funcionales	3
Autoriza Pase a QA	3
Comunicar Inicio de Pase a Producción a los usuarios	3
Ejecución de Pase a Producción	3
Pruebas de sistemas y aceptación	3
Pruebas Integrales con Sistema.	3
Solicitud de Pase a Producción – Sistema de Control de Versiones	3
Aprobación Documento de Análisis parte técnica	2
Revisión de Pase al ambiente de QA	2
Capacitación a los usuarios	1
Pruebas de comunicación	1
Pruebas de esfuerzo	1
Pruebas funcionales de flujo	1
Verificar ejecución de pase a QA/Producción	1

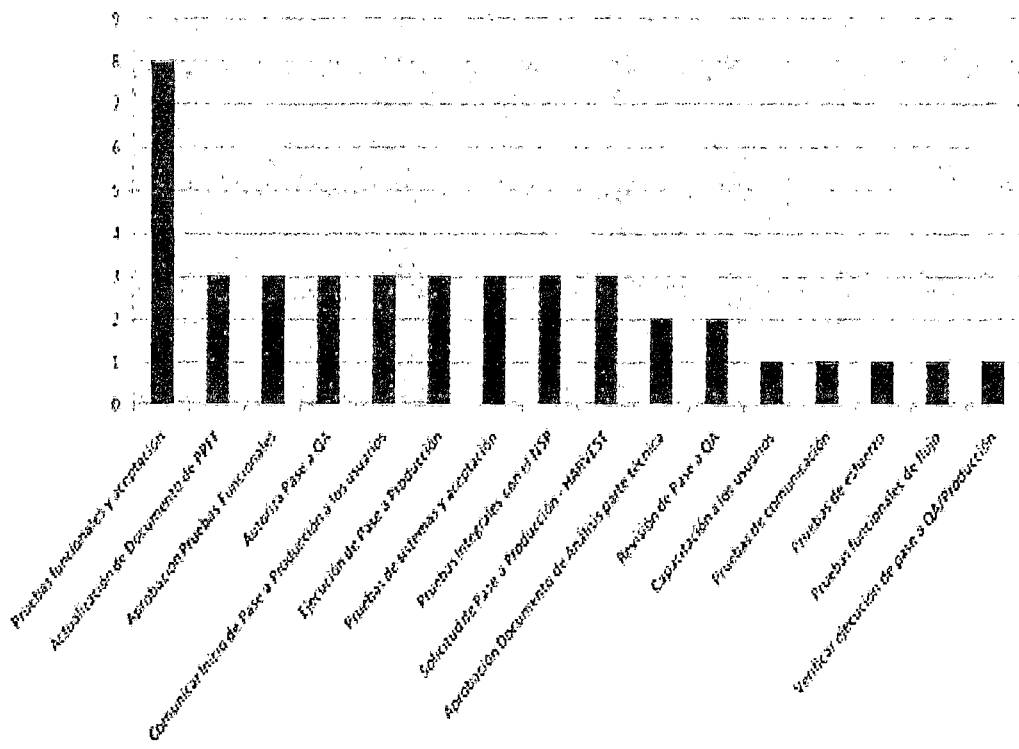


Figura 1.4: Número de errores de la actividad pruebas funcionales y de sistemas. [Guibovich, 2011]

La sub-actividad de Pruebas Funcionales es la que tiene un mayor número de iteraciones para el subproceso de Pruebas Funcionales y de Sistemas ocasionando retraso en la ejecución de las siguientes actividades.

Una manera de mejorar y poder solucionar los problemas mencionados es diseñar una metodología que combine metodologías ágiles tanto a nivel de gestión como de construcción y que permita obtener los productos de software deseables.

1.1.1 PROBLEMA GENERAL

¿Cómo diseñar una metodología de desarrollo y mantenimiento para una fábrica de software, que permitan desarrollar software con las características especificadas por el cliente?

1.1.2 PROBLEMAS ESPECÍFICOS

a) ¿Cómo identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software?
b) ¿Cómo enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo?
c) ¿Cómo diseñar la metodología que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software?
d) ¿Cómo aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software?

1.2 OBJETIVOS E HIPÓTESIS DE LA INVESTIGACIÓN

1.2.1 OBJETIVO GENERAL

Diseñar una metodología de desarrollo y mantenimiento para una fábrica de software, que permita desarrollar software con las características especificadas por el cliente.

1.2.2 OBJETIVOS ESPECÍFICOS

a) Identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software.
b) Enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo.
c) Diseñar la metodología que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software.
d) Aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software.

1.2.3 HIPÓTESIS GENERAL

El nuevo diseño de la metodología de desarrollo y mantenimiento, en la fábrica del software, permite desarrollar software con las características especificadas por el cliente.

1.2.4 HIPÓTESIS ESPECÍFICAS

a) Es posible identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software.
b) Es posible enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo.
c) Se puede diseñar la metodología, que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software.

Se puede aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software.

1.3 JUSTIFICACIÓN DE LA INVESTIGACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.3.1 IMPORTANCIA Y JUSTIFICACIÓN

La atención en forma oportuna de desarrollo y mantenimiento de sistemas de información recae en la fábrica de software, quien para poder mejorar la productividad necesita optimizar sus procesos de producción entre ellos su actual metodología de desarrollo a fin de atender la mayor cantidad de requerimientos, en un tiempo menor y de esta aumentar la satisfacción de sus clientes.

1.3.1.1 RELEVANCIA ORGANIZACIONAL

La fábrica de software requiere contar con una metodología que posibilite el desarrollo de software para satisfacer al cliente cumpliendo los niveles de servicio y calidad.

1.3.1.2 UTILIDAD METODOLÓGICA

Es estratégico que la fábrica de software cuente con una metodología que combine las mejores prácticas actuales de las metodologías de desarrollo de software para cubrir con la demanda en el mercado del software.

1.3.1.3 VALOR TEÓRICO

Mejor comprensión de la usabilidad de las metodologías de desarrollo y mantenimiento para una fábrica de software que maneja medianos y grandes proyectos desde un enfoque ágil.

1.3.1.4 IMPLICACIONES PRÁCTICAS

La fábrica de software podrá aplicar los criterios ágiles como buenas prácticas en la atención de los requerimientos orientados al desarrollo y mantenimiento de software. Asimismo, los equipos de desarrollo tendrán más tiempo para realizar tareas de análisis y mejora continua de los sistemas a mantener o desarrollar.

1.3.2 DELIMITACIÓN

El análisis se focalizó en los procesos involucrados en el Macroproceso de Realización del Producto y Servicio de la fábrica de software. La metodología de desarrollo y mantenimiento de software propuesta comprende tanto las etapas de gestión como de ingeniería de la fábrica de software; es decir, desde la recepción del requerimiento hasta la puesta en producción del software.

1.3.3 MATRIZ DE CONSISTENCIA

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	INDICADORES	ÍNDICES
PROBLEMA PRINCIPAL ¿Cómo diseñar una metodología de desarrollo y mantenimiento para una fábrica de software que permitan desarrollar software con las características especificadas por el cliente?	**OBJETIVO GENERAL** Diseñar una metodología de desarrollo y mantenimiento para una fábrica de software que permitan desarrollar software con las características especificadas por el cliente.	**HIPÓTESIS GENERAL** El nuevo diseño de la metodología de desarrollo y mantenimiento, en la fábrica del software permite desarrollar software con las características especificadas por el cliente.	**VARIABLE INTERVINIENTE** • Requerimientos de desarrollo de software. • Requerimientos de mantenimiento de software.		
			VARIABLE INDEPENDIENTE X: El uso de la metodología propuesta de desarrollo de software.	X1: Apoyo de Metodología de procesos RUP. X2: Apoyo de Metodología ágil XP en el proceso de ingeniería. X3: Apoyo de la Metodología ágil SCRUM en el proceso de gestión.	Tareas de RUP / Total de Tareas Identificadas. Tareas XP de Ingeniería / Total de Tareas Identificadas de Ingeniería. Tareas SCRUM de Gestión / Total de Tareas de Gestión Identificadas.
PROBLEMAS ESPECÍFICOS ¿Cómo identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software?	**OBJETIVOS ESPECÍFICOS** Identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica	**HIPÓTESIS ESPECÍFICAS** Es posible identificar las mejores prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica	**VARIABLE DEPENDIENTE** Y: Grado de cumplimiento de la tarea con las características especificadas.	Y1: Avance del producto software.	%APP - %ARP APP: Avance Programado del Producto Software. ARP: Avance Real del Producto Software.

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	INDICADORES	ÍNDICES
¿Cómo enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo? ¿Cómo diseñar la metodología mediante la combinación de las metodologías de desarrollo identificadas, que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software? ¿Cómo aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software?	de software. Enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo. Diseñar la metodología mediante la combinación de las metodologías de desarrollo identificadas que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software. Aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software.	de software. Es posible enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología de desarrollo. Se puede diseñar la metodología mediante la combinación de las metodologías de desarrollo identificadas, que apoye la atención de los requerimientos de desarrollo y mantenimiento de software de una fábrica de software. Se puede aplicar la nueva metodología de desarrollo a fin de demostrar que la nueva metodología permita construir un producto software.			

Tabla 1.1: Matriz de Consistencia

CAPÍTULO II

MARCO TEÓRICO Y CONCEPTUAL DE LA INVESTIGACIÓN

2.1 ANTECEDENTES DE LA INVESTIGACIÓN

Las empresas que desarrollan software, inicialmente, han trabajado siguiendo metodologías tradicionales para lograr ser más competitivas. Al pasar de los años estas metodologías ya no se amoldaban a los proyectos, es por ello que nace el paradigma ágil y las metodologías ágiles como alternativa. Por ejemplo, la *fábrica de software*, objeto de estudio en esta tesis, ha optado por mejorar sus procesos enfocados al desarrollo y mantenimiento de software y a la mejora de su metodología de desarrollo y mantenimiento. Cabe indicar, que desde su inicio 1984 hasta el 2001 se han venido aplicando las metodologías en cascada y la de *Rapid Application Development* [RAD][5]. Luego a partir del 2001 hasta el 2011 se aplicó el *Rational Unified Process* [RUP] y en el 2012 se apuesta por metodologías que incorporen métodos ágiles como el Scrum.

2.2 DESCRIPCIÓN DE LA FÁBRICA DE SOFTWARE A EVALUAR[6]

La fábrica de software nace en el Perú en 2004, como un modelo de servicios que permite gestionar el mantenimiento correctivo, evolutivo y

---

[5] El desarrollo rápido de aplicaciones o RAD [acrónimo en inglés de rapid application development] es un proceso de desarrollo de software, desarrollado inicialmente por James Martin en 1980. El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE [Computer Aided Software Engineering]. Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.

[6] No se ha considerado el nombre de la fábrica de software por confidencialidad de los resultados que se va compartir en esta tesis

desarrollo de sistemas de información. Este modelo involucra la implementación de Metodologías de Gestión, Modelos de Estimación, Métricas, Acuerdos de Niveles de Servicio y Metodología de Desarrollo de Software, para los proceso de Gestión de los Requerimientos, Planeamiento y Control de los Proyectos, Manejo de Cambios y Aceptación de los productos.

Por medio de la fábrica de software se proporcionan dos servicios. El primero, servicio de mantenimiento de aplicaciones. A través de este modelo de gestión se adaptan las prioridades del negocio definidas por el cliente. En general, este comprende el mantenimiento evolutivo y mantenimiento correctivo. El segundo, servicio de desarrollo a medida. Por medio de esta solución se desarrollan proyectos específicos de software a medida, nuevos módulos, aplicaciones nuevas y cambios en las aplicaciones actuales.

2.2.1 INTEGRACIÓN DE MODELOS DE MADUREZ DE CAPACIDADES

La fábrica de software tiene implementado hasta el nivel 3 del modelo de madurez de capacidades o *Capability Maturity Model Integration* [CMMI][7]. A continuación se muestra el esquema de la intervención del modelo CMMI en las etapas de Gestión [Soportado por la Metodología de Gestión de Proyectos] y la etapa de Ingeniería [Soportado por las áreas de proceso del CMMI nivel 3][8]

---

[7] El Modelo de Madurez de Capacidades o CMM [Capability Maturity Model], es un modelo de evaluación de los procesos de una organización. Fue desarrollado inicialmente para los procesos relativos al desarrollo e implementación de software por la Universidad Carnegie-Mellon para el SEI [Software Engineering Institute]. Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso [KPA - Key Process Area]. Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser: Definidas en un procedimiento documentado, Provistas [la organización] de los medios y formación necesarios, Ejecutadas de un modo sistemático, universal y uniforme [institucionalizadas], Medidas y Verificadas.

[8] El modelo para software [CMMI] establece 5 niveles de madurez para clasificar a las organizaciones, en función de qué áreas de procesos consiguen sus objetivos y se gestionan con principios de ingeniería. Es lo que se denomina un modelo escalonado, o centrado en la madurez de la organización. 1- Ejecutado- Inicial , 2 - Administrado - Gestionado , 3 - Definido , 4 - Administrado - Gestionado Cuantitativamente y 5 - Optimizado

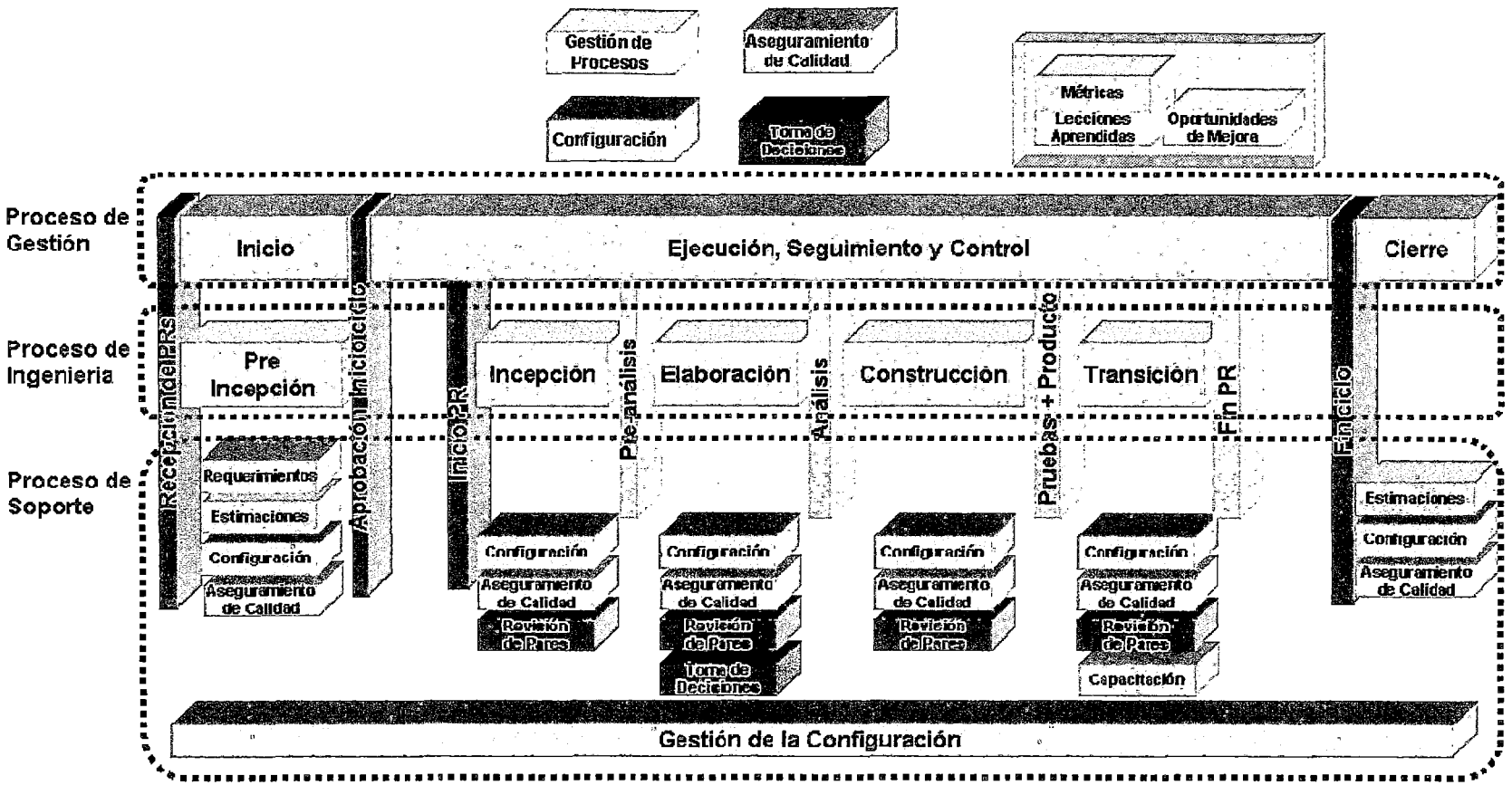


Figura 2.1: CMMI en el Proceso de Desarrollo y Mantenimiento de Software. [Manual Procesos de la Fábrica de Software, 2012]

2.3 DEFINICIONES CONCEPTUALES Y FUNDAMENTOS DE LAS VARIABLES EN ESTUDIO

2.3.1 FÁBRICA DE SOFTWARE Y BUENAS PRÁCTICAS

Según [Piattini, 2007], una fábrica de software implica una forma determinada de organizar el trabajo, con una considerable especialización así como una formalización y estandarización de los procesos.

Las fábricas de software se centran en diseños flexibles, con desarrollos dedicados a determinadas familias de producto, y grupos dedicados a desarrollar métodos y herramientas con reutilización planificada de componentes.

Una fábrica de software es una empresa de la industria del software cuya misión es el desarrollo de software para sus clientes de acuerdo a los requerimientos específicos que aquel le solicita; tiene como su principal fuente de ingreso la venta de proyectos de desarrollo de software. Generalmente, la propiedad intelectual de las aplicaciones informáticas desarrolladas le pertenece al cliente.

El modelo de fábrica de software se basa en que la empresa ofrezca servicios a la medida a sus clientes, es un concepto de subcontratación o tercerización, en el cual se delega el diseño de software a una empresa dedicada totalmente a ese fin, la cual es encargada de desarrollar plataformas para sistemas administrativos, nominas, control, procesos y mucho más.

A continuación, se resume un conjunto esencial de buenas prácticas aplicable a un amplio número de fábricas software.

2.3.1.1 INTEGRACIÓN CONTINUA Y ÉNFASIS EN LA GESTIÓN DE LA CONFIGURACIÓN

La integración continua es el proceso por el cual el producto software es integrado y/o compilado y/o desplegado así como probado y sometido a una batería de pruebas de calidad, generalmente de manera automática, con una alta frecuencia y durante la fase de desarrollo.

Esta práctica está relacionada con la gestión de la configuración, esencial en cualquier desarrollo y más aún de una fábrica de software que goza de una madurez.

2.3.1.2 CONTROL DE CALIDAD EXAHUSTIVO, PERIÓDICO Y AUTOMÁTICO

En la fabricación de software resulta imprescindible definir los objetivos y establecer mediciones.

- Fijar objetivos claros y medibles.
- Realizar las mediciones de manera periódica y frecuente.
- Medir de manera automatizada.
- Definir diferentes niveles de abstracción.

Para que estos objetivos se cumplan, las mediciones de calidad deben efectuarse durante el desarrollo a la vez que se va construyendo el producto, posibilitando una reacción rápida frente a cualquier desviación, y con la mayor frecuencia posible.

2.3.1.3 ÉNFASIS EN EL DISEÑO BASADO EN EL CONOCIMIENTO

La mayoría de decisiones en el diseño tiene un gran impacto en el resto del proyecto. Se debe ser consiente de que la calidad del diseño está asociado con la mantenibilidad y ésta con la productividad. De ello el arquitecto de software debe ser consiente y aplicar cada patrón y regla de diseño en su justa medida en función de los objetivos del negocio que se persigue.

2.3.1.4 UTILIZAR CASOS DE USO COMO PIEZA ESENCIAL DEL DESARROLLO

En la fabricación de software, los casos de uso resultan una pieza fundamental ya que permiten lograr una trazabilidad[9] efectiva desde los

[9] El término trazabilidad es definido por la Organización Internacional para la Estandarización [ISO], como la propiedad del resultado de una medida o del valor de un estándar donde éste pueda estar relacionado con referencias especificadas, usualmente estándares

requisitos al código y pueden relacionarse con varios elementos como se muestra en la Figura 2.2.

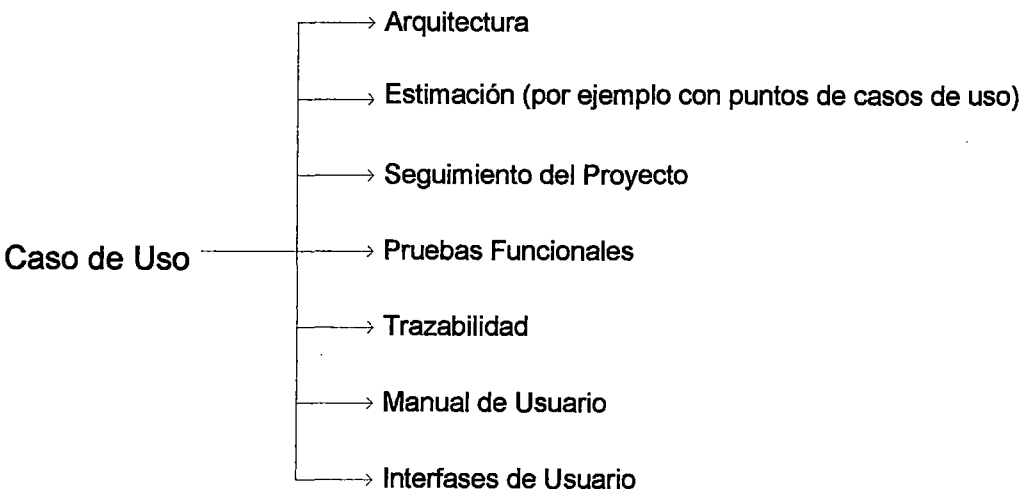


Figura 2.2: Elementos del ciclo de vida relacionados con los casos de uso. [Piattini, 2007]

2.3.1.5 CICLO DE VIDA ITERATIVO Y EVOLUTIVO

Un ciclo de vida iterativo o incremental[10], frente a un clásico en cascada, permite realizar aproximaciones que dotan de una rápida visibilidad al proyecto, a la vez que mitigan los riesgos asociados a problemas asociados con los requisitos.

---

nacionales o internacionales, a través de una cadena continua de comparaciones todas con incertidumbres especificadas.

[10] **Desarrollo iterativo y creciente** [o incremental] es un proceso de desarrollo de software, creado en respuesta a las debilidades del modelo tradicional de cascada. Permite al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior, incrementando, versiones entregables del sistema. El proceso en sí mismo consiste de:Etapa de inicialización, Etapa de iteración,Lista de control de proyecto. Para apoyar el desarrollo de proyectos por medio de este modelo se han creado frameworks [entornos de trabajo], de los cuales los dos más famosos son el Rational Unified Process y el Dynamic Systems Development Method. El desarrollo incremental e iterativo es también una parte esencial de un tipo de programación conocido como Extreme Programming y los demás frameworks de desarrollo rápido de software.

2.3.1.6 ESTIMACIÓN BASADA EN PUNTOS FUNCIÓN ADAPTADOS

Los puntos función son una métrica para establecer el tamaño y complejidad de los sistemas informáticos basada en la cantidad de funcionalidad requerida y entregada a los usuarios. Su cálculo depende en gran medida de la disponibilidad de la documentación con el suficiente nivel de detalle.

Debido a que el cálculo tradicional de los puntos función consume bastante tiempo se propone el uso simplificado, considerando que todos los elementos tienen complejidad media.

2.3.1.7 GUIAR LAS ACCIONES POR VALOR

Todas las decisiones de ingeniería, gestión y prácticas deben alinearse y medirse respecto a objetivos económicos; por ejemplo, ¿cuánto ha variado la productividad con cierta técnica de diseño?, ¿qué nivel de pruebas produce mayor retorno de inversión?, ¿qué partes del código, de ser refactorizadas[11] producen mayor valor?

2.3.2 ¿QUÉ ES UNA METODOLOGÍA?

[Maddison, 1983] [12] define **metodología** como un "*conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información*". Asimismo, [Piattini 2003] señala que una metodología es un conjunto de componentes que especifican aspectos como etapas, tareas, salidas, restricciones, herramientas y gestión.

- Cómo se debe **dividir un proyecto** en etapas.
- Qué **tareas** se llevan a cabo en cada etapa.
- Qué **salidas** se producen y cuándo se deben producir.
- Qué **restricciones**[13] se aplican.

---

[11] La refactorización [del inglés Refactoring] es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

[12] citado por [Piattini, 2003]

[13] Las restricciones son conjunto de características que limitan la metodología

- Qué **herramientas** se van a utilizar.
- Cómo se **gestiona y controla** un proyecto.

Atendiendo a una definición más genérica, se puede considerar una **metodología de desarrollo** como un conjunto de *procedimientos*, *técnicas*, *herramientas* y un *soporte documental* que ayuda a los desarrolladores a realizar nuevo software. Normalmente consistirá en un conjunto de fases descompuestas en subfases [módulos, etapas, pasos, etc.]. Esta descomposición del proceso de desarrollo guía a los desarrolladores en la elección de las técnicas que debe elegir para cada estado del proyecto, así como facilita la planificación, gestión, control y evaluación de los proyectos. *Una metodología, por tanto representa el camino para desarrollar software de una manera sistemática.*

De forma general, se puede identificar tres necesidades principales que se intentan cubrir con una metodología:

- Mejores aplicaciones. El valor de los sistemas de información resultantes dependen de multitud de pequeños factores.
- Un mejor proceso de desarrollo que identifica las **salidas** [o productos intermedios] de cada fase de forma que se pueda *planificar* y *controlar* el proyecto.
- Un proceso estándar en la organización, lo que aporta claros beneficios [por ejemplo, una mayor integración entre los sistemas y

---

Las restricciones de sistema es el eslabón más débil de la cadena que está restringiendo el throughput [volumen de trabajo o información que fluye a través del sistema] del sistema. Desde el punto de vista de un proyecto, para identificar la restricción se debe definir la sucesión de actividades más larga del proyecto que determina el tiempo del mismo; a esta sucesión se denomina la cadena crítica [las dimensiones que deben ser correctamente administradas para lograr el éxito del proyecto son cinco: equipo de desarrollo, el tiempo disponible, las funcionalidades o requisitos, el presupuesto y los recursos disponibles con que cuenta la empresa de desarrollo como el espacio físico, hardware, software y aplicativos de oficina, apoyo administrativo]. Por ejemplo en el sistema de desarrollo de software el recurso de restricción puede ser un desarrollador senior o grupo de ellos, con habilidades específicas y críticas en el proceso de desarrollo y para aprovechar sus capacidades se debe proteger que no tengan tiempo ocioso debido a la mala programación de actividades [definir una lista de actividades por cumplir identificando su criticidad]. Asimismo deben ser protegidos de interrupciones innecesarias [implementar estructuras óptimas de comunicación]. Se deben dotar de las mejores herramientas para realizar su trabajo; finalmente, es importante asegurar la buena asignación de tareas al recurso restricción con requerimientos totalmente claros y validados por los clientes / usuarios.

una mayor facilidad en el cambio de personal de un proyecto a otro].

Para [Cockburn 2000], la metodología es una convención sobre el cual un equipo está de acuerdo. La metodología se usa para los grandes problemas de coordinación de las actividades de la gente en un equipo. Un equipo de éxito depende de la cooperación, la comunicación y coordinación.

Asimismo, [Piattini 2003] menciona que las metodologías a menudo tienen distintos **objetivos** y, por lo tanto, pueden diferir una de otras. Estos pueden ser:

- Registrar los *requisitos* de un sistema de información de una forma acertada.
- Proporcionar un *método sistemático de desarrollo* de forma que se pueda controlar su progreso.
- Construir un sistema de información dentro de un *tiempo* apropiado y unos costos aceptables.
- Construir un sistema que esté bien *documentado* y que sea fácil de *mantener*.
- Ayudar a identificar, lo más pronto posible, cualquier *cambio* que sea necesario realizar dentro del proceso de desarrollo.
- Proporcionar un sistema que *satisfaga* a todas las personas afectadas por el mismo, ya sean clientes, directivos, auditores o usuarios.

Finalmente, [Fernández, 2010] define la Metodología de Desarrollo del Software [MDS], a una colección de documentación formal referente a procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una MDS es garantizar la eficacia [por ejemplo cumplir los requisitos iniciales] y la eficiencia [por ejemplo minimizar lo periodos de tiempo] en el proceso de generación del software.

2.3.3 CARACTERÍSTICAS DESEABLES DE UNA METODOLOGÍA

Según [Piattini, 2003], la metodología de desarrollo debería incluir una serie de características deseables entre las que destacan las siguientes:

- Existencia de reglas predefinidas: La metodología debería indicar formalmente unas reglas que definan sus fases, las tareas, los productos intermedios, las técnicas y herramientas, las ayudas al desarrollo y los formatos de documentación estándares.
- Cobertura del ciclo de desarrollo: Debe indicar los pasos que hay que realizar desde el planteamiento de un sistema hasta su mantenimiento, proporcionando mecanismos para integrar los resultados de una fase a la siguiente, de forma que pueda referenciar a fases previas y comprobar el trabajo realizado.
- Verificaciones intermedias: La metodología debe contemplar la realización de verificaciones sobre los productos generados en cada fase para comprobar su corrección. Se realizan por medio de revisiones de software, que detectan las inconsistencias, inexactitudes, o cualquier otro defecto que se genera durante el proceso de desarrollo, evitando que salga a relucir en la fase de pruebas, en las pruebas de aceptación, la fase de mantenimiento donde las correcciones de los defectos son más costosos que en las fases iniciales.
- Enlace con procesos de gestión: Debe proporcionar una forma de desarrollar software de una manera planificada, controlada y de calidad. Por ello si bien no se incluyen procesos de gestión de proyectos en una metodología, ésta debería dar pautas o recomendaciones para enlazar las actividades de desarrollo del proyecto con actividades propias de su gestión, como son la planificación, control y seguimiento del proyecto.

- Comunicación efectiva: Proporcionar un medio de comunicación efectiva entre los desarrollar el trabajo en grupo y con los usuarios.
- Utilización sobre un abanico amplio de proyectos: Debe ser flexible para que pueda emplearse sobre un amplio abanico de proyectos, tanto en variedad, tamaño y entorno. Una organización no debería utilizar metodologías diferentes para cada proyecto sino que la metodología se debe poder amoldar a un proyecto concreto.
- Fácil formación: La metodología la utiliza todo el personal de desarrollo de una organización, por lo que los desarrolladores deben comprender las técnicas y los procedimientos de gestión.
- Herramientas: Debe estar soportada por herramientas automatizadas que mejoren la productividad del equipo de desarrollo y la calidad de los productos resultantes. Como una metodología define las técnicas que hay que seguir en cada tarea, es necesario disponer de una herramienta que soporte la automatización de dichas tareas.
- La metodología debe contener actividades que mejoren el proceso de desarrollo: Uno de los principios de la ingeniería del software es el compromiso sobre la mejora del proceso de desarrollo. Para ello es necesario disponer de datos que muestre la efectividad de la aplicación del proceso sobre un determinado producto. Para ello es necesario definir mediciones que indique la calidad y el costo asociado a cada etapa del proceso, idealmente soportadas por herramientas. Estos datos se deben utilizar para analizar y modificar el proceso para su mejora.
- Soporte al mantenimiento: Las metodologías tradicionales se enfocaban al desarrollo de los sistemas. Pero una vez llegado

al mantenimiento no se consideraban técnicas para la evolución lógica del sistema que se encuentran en un entorno cambiante debido principalmente a los cambios tecnológicos y a las nuevas necesidades de los usuarios. La evolución del software debería ser tomada en cuenta por las metodologías para facilitar las modificaciones sobre los sistemas existentes.

- Soporte de la realización de software: Se debe incluir procedimientos para la creación, mantenimiento y recuperación de componentes reutilizables que no se limiten sólo al código.

2.3.4 CLASIFICACIÓN DE LAS METODOLOGÍAS

Para realizar la clasificación de las metodologías, se consideran tres dimensiones, como se puede ver en la Tabla 2.1. Por ejemplo, se puede considerar la metodología de análisis y diseño estructurado de [YOURDON, 1990] como una metodología de enfoque estructurado, orientada a procesos para sistemas de gestión y de tiempo real, que no utilizan métodos formales; otro ejemplo es el de la metodología de análisis y diseño de sistemas para tiempo real de [WARD y MELLOR, 1985] que se puede clasificar como una metodología estructurada orientada a procesos de tiempo real y no formal.

ENFOQUE	TIPO DE SISTEMA	FORMALIDAD
ESTRUTURADAS Orientadas a Procesos Orientadas a Datos Jerárquico No Jerárquico Mixtas	GESTIÓN	NO FORMAL
ORIENTADAS A OBJETOS	GESTIÓN / TIEMPO REAL	FORMAL

Tabla 2.1: Tres dimensiones para la clasificación de metodologías. Se pueden pensar metodologías combinando las dimensiones enfoque, tipo de sistemas y formalidad

2.3.5 PROBLEMAS DEL DESARROLLO TRADICIONAL

[Pelayo, 2007] señala que el proceso de desarrollo software de forma tradicional incluye las siguientes fases: Recogida de requisitos, Análisis, Diseño, Codificación, Prueba y Despliegue.

La Figura 2.3 esquematiza el proceso de desarrollo de software tradicional.

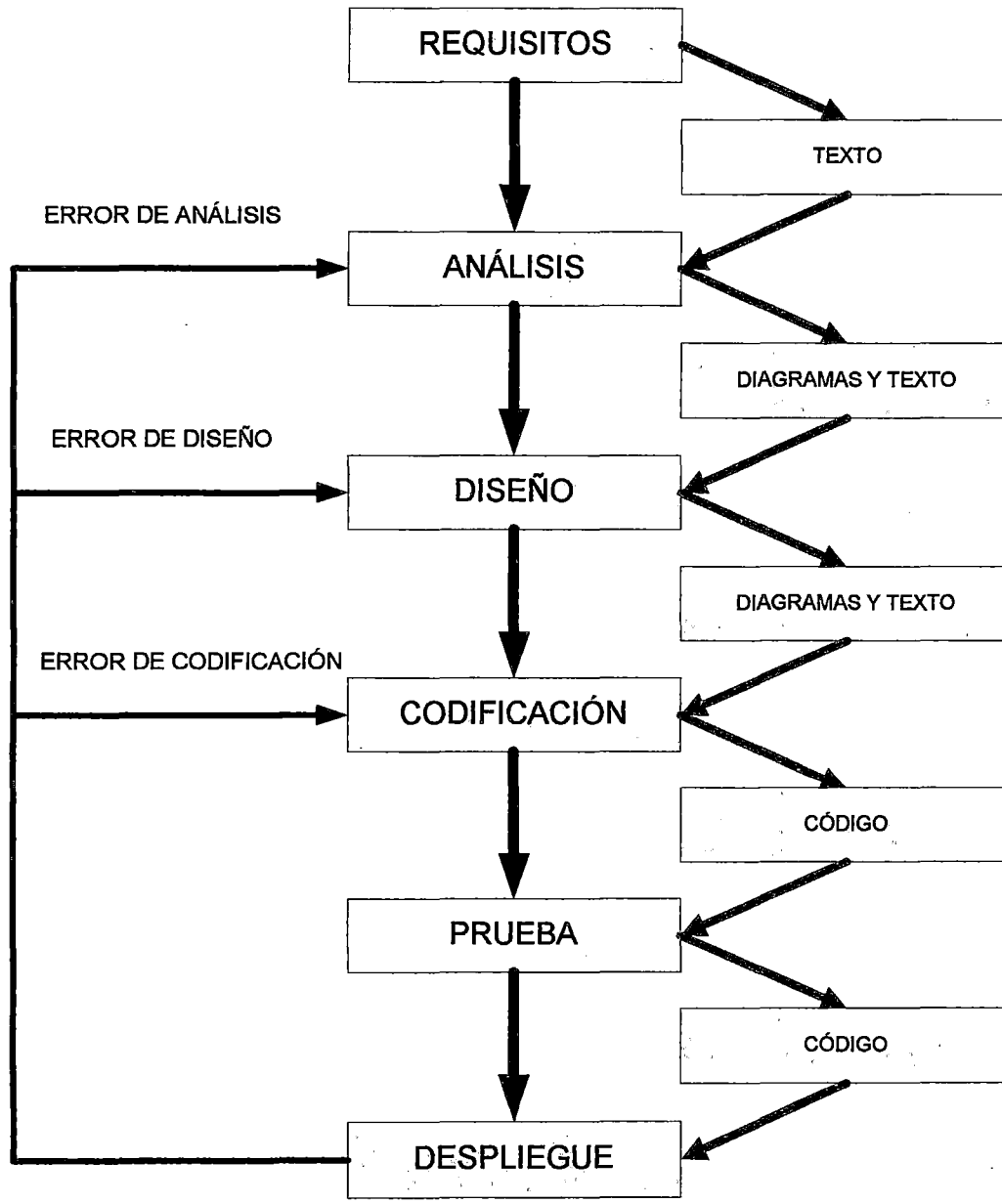


Figura 2.3: Proceso de desarrollo de Software Tradicional. [Pelayo, 2007]

Durante los últimos años se han hecho muchos progresos en el desarrollo del software, que han permitido construir sistemas más grandes y complejos. Aun así, la construcción de software de la manera tradicional sigue teniendo múltiples problemas:

2.3.5.1 PRODUCTIVIDAD

El proceso tradicional produce una gran cantidad de documentos y diagramas para especificar requisitos, clases, colaboraciones, etc. La mayoría de este material pierde su valor en cuanto comienza la fase de codificación, y gradualmente se va perdiendo la relación entre los diagramas, más aún, cuando el sistema cambia a lo largo del tiempo: realizar los cambios en todas las fases [requisitos, análisis, diseño] se hace inmanejable, así que generalmente se realizan las modificaciones sólo en el código. ¿Entonces para qué invertir tanto tiempo en construir los diagramas y la documentación de alto nivel? Lo cierto es que para sistemas complejos sigue siendo necesario. Lo que se necesita entonces es un soporte para que un cambio en cualquiera de las fases se traslade fácilmente al resto.

2.3.5.2 PORTABILIDAD

En la industria del software, cada año aparecen nuevas tecnologías y las empresas necesitan adaptarse a ellas, bien porque la demanda de esa tecnología es alta [es "lo que se lleva"], o bien porque realmente resuelve problemas importantes. Como consecuencia, el software existente debe adaptarse o migrar a la nueva tecnología. Esta migración no es ni mucho menos trivial, y obliga a las empresas a realizar un importante desembolso.

2.3.5.3 INTEROPERABILIDAD

La mayoría de sistemas necesitan comunicarse con otros, probablemente ya construidos. Incluso si los sistemas que van a interoperar se construyen desde cero, frecuentemente usan tecnologías

diferentes. Por ejemplo, un sistema que use *Enterprise JavaBeans* necesita también bases de datos relacionales como mecanismo de almacenamiento de datos. Se necesita que la interoperabilidad entre sistemas, nuevos o ya existentes, se consiga de manera sencilla y uniforme.

2.3.5.4 MANTENIMIENTO Y DOCUMENTACIÓN

Documentar un proyecto software es una tarea lenta que consume mucho tiempo, y que en realidad no interesa tanto a los que desarrollan el software, sino a aquellos que lo modificarán o lo usarán más adelante. Esto hace que se ponga poco empeño en la documentación y que generalmente no tenga una buena calidad. La solución a este problema a nivel de código es que la documentación se genere directamente del código fuente, asegurándonos que esté siempre actualizada. No obstante, la documentación de alto nivel [diagramas y texto] todavía debe ser mantenida en forma manual.

2.3.6 METODOLOGÍAS ÁGILES

Los métodos ágiles o ligeros se encuentran en contraposición a los métodos pesados tradicionales. Este tipo de métodos surgieron como respuesta al caos en el que estaba sumido el desarrollo de software, hasta el punto de que muchas veces la única planificación existente consistía en *"codificar primero y arreglar después"*. Frente a esa actitud, estos métodos trataron de imponer una disciplina en el proceso de desarrollo, con el objetivo de que éste fuese así más predecible. Para ello tomaron como fuente de inspiración otras ingenierías y crearon procesos con un fuerte componente de planificación y de documentación. Frente a estos métodos pesados, han aparecido una serie de métodos denominados ligeros o ágiles que tratan de una solución de compromiso entre la ausencia de proceso y el exceso de éste.

La diferencia más obvia entre este tipo de métodos y los pesados es que generan mucha menos documentación que aquellos. Estos métodos

ligeros están centrados en el código, de tal forma que éste se convierte en la principal documentación del proyecto. Sin embargo, no se debe considerar este aspecto como la principal diferencia entre ambos. En efecto, lo que esta ausencia de documentación pone de manifiesto son dos diferencias mucho más significativas, como afirma [Fowler, 2005]:

- Los métodos ágiles ***son adaptables, más que predictivos.*** Los métodos pesados tienden a planificar con gran detalle todo el proceso de desarrollo, pero esto sólo funciona mientras las cosas no cambien. Frente a esta resistencia al cambio, los métodos ágiles asumen que éste se va a producir, y se preparan para recibirlo.
- Los métodos ágiles se centran más en las personas que en el proceso. A nuestro juicio, ésta constituye una de las principales características de este tipo de métodos, que explicitan así la necesidad de tener en cuenta la naturaleza de las personas en vez de ir contra ella.

Las características de estos métodos se muestran en la Tabla 2.2 .Estas características están extraídas del Manifiesto para el Desarrollo de Software Ágil [Beck, 2001], firmado en febrero de 2001 por representantes de Programación Extrema, Scrum, DSDM y otros métodos ágiles.

Se prefiere	Frente a
Las personas y las relaciones	Los procesos y herramientas
El software que funciona	La documentación
La colaboración del cliente	Los contratos
Responder a los cambios	Seguir un plan

Tabla 2.2: Características de los métodos ágiles

2.3.7 FACTIBILIDAD Y APLICABILIDAD DE LA METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE SOFTWARE

Según la metodología *Dynamic System Development Method* [14][DSDM] una de las fases de la construcción del sistema es el estudio de la factibilidad, el cual determina si la metodología se ajusta al proyecto.

Para resolver la aplicabilidad de un proyecto, *DSDM* plantea las siguientes preguntas:

- ¿Será la funcionalidad razonablemente visible en la interfase del usuario?
- ¿Se pueden identificar todas las clases de usuarios finales?
- ¿Es la aplicación computacionalmente compleja?
- ¿Es la aplicación potencialmente grande? Si lo es, ¿puede ser particionada en componentes funcionales más pequeños?
- ¿Está el proyecto realmente acotado en el tiempo?
- ¿Son los requerimientos flexibles y sólo especificados a un alto nivel?

[Leffingwell, 2007] menciona que RUP es un marco de trabajo de proceso de software en la que se puede crear varias instancias del RUP para proyectos específicos. Así, RUP puede ser aplicado para cualquier tipo de proyecto de software de diferentes alcances [grande, mediano y pequeño]; pero RUP, se orienta más hacia construcciones de sistemas altamente complejos[15].

Por otro lado, respecto al XP [Leffingwell, 2007] se refiere a que los límites de XP no son claras todavía. Se han identificado tipos de

---

[14] El método de desarrollo de sistemas dinámicos [en inglés Dynamic Systems Development Method o DSDM] es un método que provee un framework para el desarrollo ágil de software, apoyado por su continua implicación del usuario en un desarrollo iterativo y creciente que sea sensible a los requerimientos cambiantes, para desarrollar un sistema que reúna las necesidades de la empresa en tiempo y presupuesto. Como extensión del Desarrollo rápido de aplicaciones [RAD], DSDM se centra en los proyectos de sistemas de información que son caracterizados por presupuestos y agendas apretadas.

[15] Un sistema complejo se compone de subsistemas relacionados que tienen a su vez sus propios subsistemas, y así sucesivamente hasta que se alcance algún nivel ínfimo de componentes elementales.

proyectos en las cuales la aplicación de XP no es adecuado, las cuales son:

- Cualquier proyecto en la que el costo del cambio se incrementa exponencialmente con el tiempo.
- Sistemas críticos de seguridad.
- Proyectos en las cuales culturalmente es inaceptable desarrollar sin documentos formales de documentos de: requerimientos, análisis y diseño o proyectos que son dirigidos por el formalismo.
- Proyectos o compañías en las cuales no es aceptado trabajar solo 40 horas por semana. Ya que para XP la productividad decrece cuando hay un incremento de sobretiempo.
- Proyectos en la que la integración continua, las pruebas continuas no son factibles.
- Plantas físicas. XP depende de pequeños grupos de equipos juntos. Poner dos programadores trabajando lo mismo pero en diferentes pisos violaría el principio de XP.

En cuanto a Scrum, [Leffingwell, 2007] menciona que ha sido exitosamente aplicado en miles de proyectos a nivel mundial, pero así como XP no puede ser aplicado por todos. Scrum no se recomienda para los siguientes casos:

- Equipos grandes y distribuidos. Como el XP, Scrum trabaja como máximo con 8 personas.
- Culturas en las que el equipo no se le permite tener poder de decisión.
- Proyectos en la que la integración continua y el testeo no es factible.

De acuerdo a [Núñez, 2010], para realizar la adaptación de metodologías incluye considerar criterios de comparación, a partir de los cuales permitirá decidir por una de las metodologías. Asimismo, señala que la adopción de una o varias metodología tiene sentido si se establece un caso lo suficientemente concreto, invalidando cualquier tipo de

generalización y por ende una metodología se hace necesario para cada solución en base a las necesidades que tenga un proyecto. Por ejemplo, si el proyecto necesita pautas claras de gestión se debería seleccionar Scrum y AUP[16], si el proyecto presenta ratios de error elevados, se puede utilizar XP o AUP, debido a que ambas metodologías hace uso del *Test Driven Development* [TDD], o quizás si es un proyecto interactivo con la funcionalidad visible en la interfase de usuario, se debería optar por DSDM. Por ejemplo, para el caso de la metodología propuesta se combinaron tres metodologías, con el objetivo de dar una cobertura ágil a un ámbito más amplio del ciclo de vida del software. Las metodologías seleccionadas son:

- Scrum, para la parte de gestión del proyecto.
- XP, por ser una metodología que cubre las actividades desde el plano completo de la ISO 12207 [17], perteneciente al desarrollo de software.
- RUP, por ser una metodología que contiene actividades que se requieren para ejecutar proyectos grandes.

2.4 BASES TEÓRICAS

2.4.1 DESARROLLO Y MANTENIMIENTO/EVOLUCIÓN DEL PRODUCTO SOFTWARE

La ISO/EIC 14764 [1999] define mantenimiento del software[18] como el proceso por el cual un producto sufre modificaciones en su código y documentación asociada para solucionar un problema o mejorar. El estándar 1219 de IEEE [1998] lo define como la modificación de un producto software tras ser liberado para corregir defectos, mejorar la

---

[16] El Proceso Unificado Agil de Scott Ambler o Agile Unified Process [AUP es una versión simplificada del Proceso Unificado de Rational [RUP]. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas [test driven development - TDD], Modelado Agil, Gestión de Cambios Agil, y Refactorización de Base de Datos para mejorar la productividad.

[17] Es el estándar para los procesos de ciclo de vida del software de la organización ISO.

[18] Para simplificar, en este estudio consideraremos "mantenimiento" y "evolución" como sinónimos referidos a todo el trabajo ejecutado tras la primera release del software.

ejecución de otros atributos, o adaptar el producto a un nuevo entorno. Sea cual sea la definición que se considere, el concepto de evolución del software implica un cambio continuo desde un estado menor, más simple o peor a uno superior o mejor. Los objetivos que se persiguen al evolucionar un producto software son diversos: mantener operativo el sistema, incrementar la satisfacción de los usuarios, maximizar la inversión y reducir los costos, alargar la vida del software o adaptarlo a nuevos cambios o requisitos. Las conocidas **Leyes de Lehman** [Lehman, 1997] para la evolución del software establecen siete leyes:

Primera Ley: CAMBIO CONTINUO. Un programa grande que es utilizado se somete a un cambio persistente o se convierte en menos útil progresivamente. El proceso de cambio continúa hasta que se juzga como más rentable reemplazar el sistema por una nueva versión. Un programa destinado a solucionar un problema del mundo real que se utiliza, debe adaptarse continuamente, en caso contrario, el programa se hace progresivamente menos satisfactorio. Estas adaptaciones son el resultado del cambio en la operación del entorno en el cual la aplicación cumple una función.

Segunda Ley: COMPLEJIDAD CRECIENTE. Como un programa grande cambia de forma continua, su complejidad la cual refleja una estructura deteriorándose, se incrementa a menos que se realice un trabajo para mantenerla o reducirla. Estas leyes no son otra cosa que el resultado del estudio científico de experiencia acumulada en Ingeniería del Software. Como tales, nos pueden servir como base para la planificación de las actividades de mantenimiento y para la toma de decisiones al respecto. A medida que evoluciona un programa, su complejidad se incremente, a menos que se trabaje para mantenerla o reducirla. Esta ley implica un tipo de "degradación" o "entropía19" en la estructura del programa. Esto a su vez implica un aumento progresivo del

[19] Magnitud termodinámica que indica el grado de desorden molecular de la materia.

esfuerzo de mantenimiento, a menos que se realice algún tipo de mantenimiento perfectivo a este respecto.

Tercera Ley: AUTORREGULACIÓN. Las medidas de un proyecto global y atributos del sistema se auto regulan cíclicamente con tendencias e invariantes estadísticamente determinables. El proceso de evolución del programa se autorregula con una distribución de medidas de atributos de producto y procesos cercana a la normal. La evolución de programas industriales programa destinados a solucionar un problema del mundo real, se lleva a cabo por un equipo que opera en una organización más grande. Las decisiones de gestión respecto a los cambios en el programa constituyen una dinámica que determina las características de crecimiento del producto.

Cuarta Ley: CONSERVACIÓN DE LA ESTABILIDAD ORGANIZATIVA [VELOCIDAD DE TRABAJO INVARIANTE]. El ratio de actividad global en un gran proyecto de programación es invariante. La velocidad de actividad global efectiva media en un sistema en evolución es invariante a lo largo del ciclo de vida del producto. Usualmente se considera que el esfuerzo gastado en la evolución del sistema se determina por decisiones de dirección. Esto es por supuesto así en un cierto grado, pero su influencia está limitada por factores externos respecto al empleo, la disponibilidad de personal competente, etc. No obstante, también influyen los atributos del sistema, por ejemplo, la complejidad. Los datos empíricos sugieren que la actividad lleva a una estabilización de actividad aproximadamente constante.

Quinta Ley: CONSERVACIÓN DE LA FAMILIARIDAD. En una evolución fiable y planificada, un gran programa sometido a un cambio debe estar disponible para una ejecución regular del usuario en el máximo intervalo determinado por su crecimiento neto. Es decir, el sistema desarrolla un promedio característico de crecimiento seguro, que de ser excedido, causa problemas de calidad y utilización con tiempo y costo que excede del previsto. Durante la vida activa de un programa en evolución, el contenido de las versiones sucesivas es estadísticamente invariante.

Uno de los factores que determina el progreso de un desarrollo de software es la familiaridad de todos los implicados. Cuantos más cambios y adiciones se hacen a una versión, es más difícil que todos los implicados la conozcan. Debido a que el crecimiento está limitado por la capacidad de adquirir información de los participantes, una evolución "grande" dificultaría ese aprendizaje, por lo que los cambios tienden a ser de un tamaño parecido y limitado.

Sexta Ley: CRECIMIENTO CONTINUO. El contenido funciona de un programa debe incrementarse continuamente para mantener la satisfacción del usuario durante su ciclo de vida. Esta ley refleja un aspecto del mismo fenómeno que refleja la primera. Habitualmente, los sistemas se crean con una limitación en cuanto a la funcionalidad del dominio cubierta, por motivos de tiempo o recursos. Esto hace que con el tiempo, los requisitos que se descartaron vuelvan a aparecer como necesidades.

Séptima Ley: CALIDAD DECRECIENTE. Los programas destinados a solucionar un problema del mundo real serán percibidos como de calidad decreciente a menos que se mantengan de manera rigurosa y se adapten al entorno operativo cambiante. Esta percepción de la calidad decreciente tiene que ver con los cambios en los criterios de aceptabilidad de los usuarios.

Asimismo, se debe considerar que en el ciclo de vida del desarrollo de software, la primera versión es sólo una pequeña parte del trabajo ya que el mantenimiento y la evolución cubren la mayoría del ciclo de vida. De hecho, se estima que la evolución del software consume un 80% del presupuesto mientras que el desarrollo es el 20% restante[20]. La evolución del software implica un mantenimiento del mismo durante el desarrollo pero se convierte en una garantía frente a la obsolescencia. El turbulento entorno que rodea la producción software en la actualidad, y el cambio

[20] Ley de Pareto

continuo en las necesidades del cliente implican un proceso evolutivo constante.

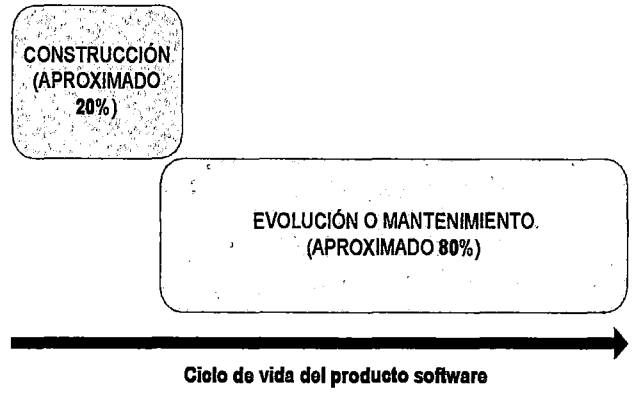


Figura 2. 4: Esfuerzo del mantenimiento en el ciclo de vida de un producto. software. [Rodríguez, 2008]

2.4.1.1 EL PROCESO DE EVOLUCIÓN DEL PRODUCTO SOFTWARE

Desde una perspectiva convencional la evolución del producto software sigue un rígido proceso, cuyo inicio es la gestión de cambios, que permite controlar las distintas versiones y cambios de un sistema durante su ciclo de vida. En la gestión de cambios, las peticiones son registradas y se pasa a una fase de análisis del impacto en la que se determina el ámbito de los cambios requeridos como base para su implementación. Las actividades principales de esta fase son:

- Evaluar las peticiones de cambios sobre los sistemas existentes, otros sistemas, hardware, estructuras de datos, etc.
- Desarrollar una estimación preliminar de recursos.
- Documentar el ámbito del cambio y actualizar la petición realizada.

Una vez realizado el análisis de impacto se planifica la versión del sistema para determinar el contenido y tiempo de las versiones del sistema, constituyendo un subproceso en sí mismo con un amplio conjunto de actividades:

- Ordenar y seleccionar las peticiones de cambio para la próxima versión.
- Procesar los cambios y organizar el trabajo.
- Preparar un documento sobre el plan de la versión del sistema.
- Actualizar las peticiones de cambio aprobadas.

Constituida esta etapa, que se puede considerar como pre-evolución, se pasa a la fase de evolución en sí del producto. Se comienza introduciendo cambios en el diseño consistentes en revisar el diseño lógico [nivel de sistema] y físico [nivel de programa] para los cambios aprobados. Las principales actividades en esta fase son:

- Analizar los cambios aprobados y revisar la estructura del programa.
- Revisar o desarrollar los diseños lógicos y físicos.
- Diseñar los cambios de hardware si fueran necesarios.
- Actualizar los documentos del sistema y del programa.
- Restaurar los documentos bajo control del sistema de gestión de cambios.
- Actualizar las peticiones de cambios para reflejarlos en los documentos.

Posteriormente se pasa a la fase de codificación con el fin de reflejar los cambios aprobados y representados en diseño. Esta tarea implica:

- Implementar y revisar los cambios en el código.
- Restaurar el código fuente bajo control del sistema de gestión de la configuración.
- Actualizar las peticiones de cambios de los módulos modificados.

Y, finalmente, se ejecutan las pruebas o test tanto para asegurar la conformidad de los cambios aprobados con los requisitos originales como para asegurar la calidad del producto final. Esta fase es muy importante ya que los resultados obtenidos dependerán, en gran medida, de un

correcto diseño de pruebas. Un aspecto a destacar en este sentido es que sólo necesitan revisarse los cambios y no el producto concreto.

2.4.1.2 TIPOS DE EVOLUCIÓN/MANTENIMIENTO

En función del objetivo, existen cuatro tipos de mantenimiento o evolución del software que a continuación se detallan.

2.4.1.2.1 EVOLUCIÓN CORRECTIVA

Se define como aquel proceso orientado a la reparación de defectos existentes en un sistema software. Se utiliza, por ejemplo, cuando un programa falla o aborta, o cuando produce resultados que no son acordes con los requisitos o el diseño. Este tipo de evolución tiene asociada dos problemas principales:

- Por un lado, reparar un defecto tiene la probabilidad de introducir otro defecto.
- Por otro lado, incrementa notablemente las operaciones de testeo.

La evolución correctiva se suele utilizar bien en reparaciones de emergencia, ejecutadas en cortos periodos de tiempo y, generalmente, sobre un único programa, o bien, en reparaciones planificadas que arreglan defectos que no requieren una atención inmediata.

2.4.1.2.2 EVOLUCIÓN ADAPTATIVA

Se define como el proceso para mejorar la funcionalidad del software, hardware y su documentación. El mantenimiento adaptativo está formado por cinco pasos:

- Definición de requisitos, donde se revisan y entienden los requisitos de nuevos cambios.
- Diseño del sistema, para determinar dónde y cuándo implementar los cambios en el sistema.
- Diseño de datos, ya que los cambios en las estructuras de datos también deben ser diseñados.

- Diseño del programa, para analizar los documentos de diseño del programa y determinar dónde añadir, modificar y borrar las funciones que implementan los cambios propuestos.
- Diseño del módulo, donde se analizan los documentos que determinan cómo integrar cambios en un módulo existente o bien crear uno nuevo.

La evolución objeto de esta investigación se encuadra en este grupo. Los beneficios de este tipo de evolución proporciona son muy significativos. Entre otras, se puede destacar que mejora la productividad automatizando actividades, mejora de respuestas a oportunidades del negocio.

2.4.1.2.3 EVOLUCIÓN PERFECTIVA

Es un método para tratar de pulir o refinar la calidad del software y su documentación. El objetivo principal es hacerlo más fácilmente mantenible reduciendo el costo e impacto de futuros cambios. Para cumplir este objetivo se llevan a cabo actividades de reingeniería, reescritura y actualización de la documentación.

2.4.1.2.4 EVOLUCIÓN PREVENTIVA

Es la que se ejecuta para prevenir fallos antes de que éstos ocurran. Por ejemplo, un chequeo de tipos antes de compilar y ejecutar una aplicación. Las técnicas más importantes para este tipo de evolución son:

- Análisis de complejidad, para medir la complejidad de los módulos.
- Análisis de funcionalidad, ya que medir la funcionalidad de un sistema ayuda a estimar su valor.
- Ingeniería inversa, consistente en recuperar el diseño cuando el código no es acorde con la documentación.
- Reingeniería por partes, para sistemas de vida corta y cambios evolucionarios necesarios.
- Translación/reestructuración/modularización, para migraciones de plataforma.

2.4.1.3 ESTUDIOS EMPÍRICOS SOBRE METODOLOGÍAS ÁGILES Y EVOLUCIÓN DEL SOFTWARE

Desde hace décadas, muchos estudios han centrado sus esfuerzos en estudiar la evolución del software en metodologías convencionales. Está constatado que los cambios acumulados y el crecimiento de la complejidad es inevitable en software evolucionado. También se ha estudiado la necesidad de realizar trabajos extras con el objetivo de disminuir esta complejidad [refactorización].

Por otro lado, el amplio proceso que requiere la evolución de software en metodologías convencionales descrito no se ajusta a los valores en los que se basan las metodologías ágiles. Sin embargo, considerando los principios en los que las sustentan como entregas continuas de software que funcione y aporten valor al cliente y adaptación a las variaciones en las necesidades del cliente y, por tanto, en los requisitos que debe cumplir el sistema, se puede deducir que la evolución del código es un objetivo explícito de las metodologías ágiles y la refactorización es vista como algo necesario y positivo.

La atención de los estudios sobre metodologías ágiles realizados hasta el momento se centra en la situación antes y durante el desarrollo inicial. Por ejemplo, [Boehm y Turner, 2005] establecen cinco dimensiones de riesgo a considerar para decidir la metodología a utilizar [tamaño, criticidad, dinamismo, equipo y cultura], sin considerar el tipo de desarrollo. Evidencias empíricas en este sentido podrían ayudar a las organizaciones a decidir qué método escoger.

Sorprendentemente, a pesar de que la mayor parte del desarrollo software se centra en mantener y evolucionar productos, pocos estudios se han dedicado a estudiar la evolución y mantenimiento del software ágil, ya que requiere tanto de observaciones cualitativas y cuantitativas, cuya recolección y estudio es dificultoso. Una de las pocas excepciones fue el estudio desarrollado por [Chapin,2003] donde discute cómo encajan los procesos de agilidad en la evolución del software considerando los

diferentes tipos de stakeholders[21] y concluyendo que los efectos de utilizar metodologías ágiles pueden ser más positivos para unos que para otros. [Wernick y Hall, 2001] argumentan que la técnica de *programación en pareja* podría ser beneficiosa cuando la evolución es prolongada en el tiempo. Otro estudio destacable es en el que se presentan los resultados del estudio llevado a cabo en una organización donde un método ágil fue introducido en un entorno de desarrollo de evolución y mantenimiento de producto software, aunque se trata de un amplio estudio que no se centra explícitamente en la característica de evolución.

Por último, [Pries-Heje y Lindwall, 2001] divulgan los resultados obtenidos al realizar entrevistas en nueve organizaciones en desarrollo software sobre la evolución de productos utilizando metodologías ágiles. Intentan averiguar cuál es el motivo por el cual, a pesar de que la evolución del software y las metodologías ágiles están tan relacionadas, son conceptos disjuntos en las mayoría de las organizaciones. La conclusión a la que llegan es que en las organizaciones el proceso de mantenimiento y evolución del software sigue caminos muy tradicionales, puesto que el pilar que guía la evolución es la calidad y en ciertos sectores de la ingeniería del software se tiene la idea de que las metodologías ágiles son un proceso de codificación, sin documentación ni control alguno que no asegura la calidad de los productos. Por lo tanto, concluyen que se trata de un área en el que las metodologías ágiles aún no han sido altamente introducidas. Finalmente, solicitan la realización de estudios en este sentido que ahonden sobre los beneficios que las metodologías ágiles pueden proporcionar en la evolución y mantenimiento de productos software.

Sin embargo, las conclusiones obtenidas en estos estudios tienen carácter cualitativo. Tras una ardua exploración del estado de la

---

[21] La definición más correcta de "stakeholder" sería parte interesada [del inglés stake, apuesta, y holder, poseedor]. Se puede definir como cualquier persona o entidad que es afectada o concernida por las actividades o la marcha de una organización; por ejemplo, los trabajadores de esa organización, sus accionistas, las asociaciones de vecinos afectadas o ligadas, los sindicatos, las organizaciones civiles y gubernamentales que se encuentren vinculadas, etc.

investigación en este campo, desde nuestro conocimiento, solamente se presenta un estudio basado en medidas. La experiencia de este estudio reporta que las metodologías ágiles permiten una evolución aboliendo problemas de incremento de la complejidad del código o decremento en la satisfacción de los clientes. Implícitamente, solicita el desarrollo de más sistemas utilizando metodologías ágiles para estudiar si las conclusiones obtenidas pueden ser generalizadas.

2.4.2 REQUERIMIENTOS DE LA FÁBRICA DE SOFTWARE

Los requerimientos se definen como características que se desea que posea un sistema o un software. La ingeniería de requisitos cubre las tareas relacionadas con los requerimientos de un sistema. Normalmente, un tema de la Ingeniería de Software tiene diferentes significados. De las muchas definiciones que existen para requerimiento, a continuación se presenta la definición que aparece en el glosario de la IEEE[22].

1. Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
2. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
3. Una representación documentada de una condición o capacidad como en [1] ó [2].

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas; de otro lado, los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento [en tiempo y espacio], interfaces de usuario,

[22] IEEE Std. 610.12-1990

fiabilidad [robustez del sistema, disponibilidad de equipo], mantenimiento, seguridad, portabilidad, estándares, etc.

2.4.2.1 CARACTERÍSTICAS DE LOS REQUERIMIENTOS

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. A continuación se presentan las más importantes.

- Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.
- Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.
- Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
- Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.
- No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
- Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

2.4.2.2 DIFICULTADES PARA DEFINIR LOS REQUERIMIENTOS

- Los requerimientos no son obvios y vienen de muchas fuentes.

- Son difíciles de expresar en palabras [el lenguaje es ambiguo].
- Existen muchos tipos de requerimientos y diferentes niveles de detalle.
- La cantidad de requerimientos en un proyecto puede ser difícil de manejar.
- Nunca son iguales. Algunos son más difíciles, más riesgosos, más importantes o más estables que otros.
- Los requerimientos están relacionados unos con otros, y a su vez se relacionan con otras partes del proceso.
- Cada requerimiento tiene propiedades únicas y abarcan áreas funcionales específicas.
- Un requerimiento puede cambiar a lo largo del ciclo de desarrollo.
- Son difíciles de cuantificar, ya que cada conjunto de requerimientos es particular para cada proyecto.

2.4.3 DESCRIPCIÓN DE LA METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE LA FÁBRICA DE SOFTWARE Y LIMITACIONES

Los procesos y subprocesos que conforman el Desarrollo y Mantenimiento de Software de la fábrica de software, objeto de estudio de esta Tesis, se presenta a continuación.

1. Proceso de Mantenimiento de Software	
1.1. Etapa Gestión	
1.1.1. Inicio y Planificación	En esta etapa se definen las actividades de planificación del ciclo de producción de mantenimiento. Cada línea de producción elabora la Lista Maestra de Requerimientos de los Requerimientos, los estima con la Herramienta de Estimación y elabora el Cronograma de trabajo para la línea de producción. De ser necesario se elabora un análisis preliminar, el Analista de Sistemas asignado para la atención del requerimiento es el responsable de convocar a una reunión al Usuario

	autorizado por el Cliente en la cual establecen el alcance del requerimiento, evalúan el impacto en el aplicativo y procesos involucrados, el Analista de Sistemas estima de manera preliminar el tiempo de atención. Las principales actividades son. Entregar reporte de Requerimientos priorizados, elaborar plan de mantenimiento, asignar requerimientos, elaborar LMR y estimar tiempos con recursos, elaborar cronograma, revisar cronograma, enviar documento a cliente, revisar documento, enviar documento a calidad, grabar línea base y actualizar la matriz de cambio, notificar al Gestor de Configuración [GC] la aprobación del cronograma, etiquetar cronograma y grabar línea base.
1.1.2. Ejecución, Seguimiento y Control	En esta etapa se asigna y ejecuta el trabajo acordado con el cliente para el ciclo de producción. Se realiza el seguimiento a los planes, problemas, riesgos entre otros temas, en los comités de seguimientos. Se procesan todos los cambios que se presenten durante la ejecución del ciclo de producción. Las principales actividades son: asignar trabajo, generar informe de estado, recolectar métricas, realizar comité interno de línea, elaborar acta, realizar comité de analistas, realizar comité de sistemas, actualizar cronograma.
1.1.3. Cierre	En esta etapa se elabora el Informe de Cierre de Ciclo de Producción. Se elabora y revisa el relatorio del ciclo de producción. Se archivan los registros generados durante el ciclo de producción. Las principales actividades son: elaborar informe de cierre de ciclo, elaborar relatorio, mantener parámetros de configuración, revisar QA, ejecutar gestión de la configuración.
1.2. Etapa Ingeniería	
1.2.1. Incepción y Elaboración	Se elabora el documento de Pre-Análisis y de Análisis. Las principales actividades son: registrar, priorizar y asignar los requerimientos; realizar el documento de pre-análisis y de análisis, auditaría por Gestión de Configuración, enviar documento al cliente, revisar el documento.
1.2.2. Construcción	Se implementa la solución y se desarrollan las pruebas funcionales y de sistemas. Se elabora el documento de Pase a QA/Producción. Las principales actividades son: implementar modificaciones, realizar pruebas unitarias, elaborar casos de prueba, auditoría por Gestión de Configuración, elaborar casos de prueba, integrar el producto, levantar observaciones, realizar pruebas internas, elaborar documento de pase a QA, solicitar

	pruebas de calidad, realizar pruebas de QA, solicitar pase a QA, realizar pruebas funcionales y de sistemas, solicitar pase a producción.
1.2.3. Transición	Llevar la solución implementada al ambiente de producción y actualizar los manuales de usuario, de sistemas y de administración de sistemas. Las principales actividades son: ejecutar pase a producción, verificar pase, revisión y aprobación de documentación.
2. Proceso de Desarrollo de Software	
2.1. Etapa Gestión	
2.1.1. Elaboración propuesta	En esta fase debe consolidarse la estimación y el alcance del proyecto especial. Esta actividad sólo se realiza en caso de no tener el documento "Plan de Proyecto Inicial" aprobado por el cliente. Las principales actividades son: solicitar desarrollo, evaluar condiciones, elaborar plan de elaboración de propuesta, elaborar LMR y WBS, elaborar cronograma, completar plan de proyecto inicial, aprobar plan, elaborar Kick Off Meeting.
2.1.2. Inicio	Esta etapa se realiza de forma paralela a la fase de incepción. [Ingeniería]. En esta etapa se complementa el Plan de Proyecto con todos los planes de soporte relacionados al proyecto. Adicionalmente se actualiza la fecha de inicio del proyecto [en caso haya cambiado] y el WBS del proyecto. Realizada la aprobación del Plan de Proyecto se procede a realizar las presentaciones de Kick Off, tanto interno como externo. Las actividades principales son: actualizar plan de proyecto, revisar plan de proyecto, ejecutar reunión interna de inicio del trabajo, ejecutar reunión externa de inicio del proyecto.
2.1.3. Ejecución, Seguimiento y Control	En esta etapa, se ejecuta el "Plan del Proyecto" y se realizan las actividades de seguimiento sobre lo planificado, se asigna actividades al equipo de trabajo. Se realiza el control de cambios al Plan de Proyecto. Los riesgos son revisados en los comités definidos y escalados de forma jerárquica. En el tablero de seguimiento de pendientes se registra constantemente los ítems de acción relevantes para realizar un seguimiento. La matriz de entregables es actualizada mensualmente a fin proporcionar información del estado de los entregables principales a la jefatura del servicio. Las principales actividades son: realizar comité interno, asignar trabajo, realizar comité de analistas, realizar comité de sistemas.
2.14. Cierre	Realiza el Informe de Cierre del proyecto. Se redacta el acta de cierre de proyecto, el acta debe ser aprobada por el cliente. Se registra las oportunidades de mejora y las lecciones aprendidas. Las principales actividades son: elaborar informe de monitoreo, elaborar informe de final de proyecto y actas de cierre, elaborar el relatorio del proyecto.

2.2. Etapa Ingeniería	
2.2.1. Pre - Incepción e Incepción	Esta etapa es conocida como la etapa de elaboración de propuesta, en la cual se debe realizar las actividades necesarias para determinar el alcance del proyecto y poder plasmarlo en el Plan de Proyecto. El Plan de Proyecto debe contener la estimación de todo el proyecto, en base al análisis del Modelo de Negocio y al documento de Alcance del Proyecto. Así mismo, se debe completar el Documento de Alcance del Proyecto, lo cual será validado en conjunto con el Plan de Proyecto Completo. Las principales actividades son: elaborar modelo de negocio, auditoría por gestión de la configuración, aprobar modelo de negocio, elaborar y priorizar el inventario de casos de uso, realizar análisis de alternativas de solución, elaborar plan de migración y/o documento de alcance, realizar revisión par, elaborar documento de alcance.
2.2.2. Elaboración	Los documentos que deben ser aprobados por el cliente en esta fase son: Documento de Análisis, Documento de Diseño, Documento de Implementación y Plan de Migración. Las principales actividades son: elaborar documento de análisis, construir prototipos de interfaces de usuario, enviar documentos a revisión, auditoría de gestión de la configuración, definir arquitectura del negocio, elaborar documento de implementación, elaborar plan de iteraciones, elaborar documento de diseño, elaborar modelo de datos, elaborar diccionario de datos, revisión de calidad.
2.2.3. Construcción	La finalidad principal es completar el desarrollo del sistema basado en la arquitectura de línea base. En esta fase se desarrolla el software y se realizan las pruebas respectivas de validación. Las principales actividades son realizar implementación, diseñar plan de pruebas, elaborar casos de prueba, elaborar plan de capacitación y de implementación, realizar pruebas internas, elaborar y enviar documento de pase a QA, realizar pruebas de calidad, solicitar pase a QA, realizar pruebas de sistema funcionales, elaborar informe de pruebas, capacitar a usuarios, solicitar pase a producción.
2.2.4. Transición	En esta etapa, el personal de la fábrica de software actúa como soporte y actualiza la documentación. Las principales actividades son: ejecutar pase a producción, actualizar manuales, elaborar informe de capacitación.

Fuente: [Guibovich, 2011]

Dentro de las limitaciones de la metodología de la fábrica de software se puede mencionar lo siguiente:

- En cuanto al tamaño, es extensivo y de difícil comprensión.

- Alta complejidad y gran amplitud que requiere su personalización para ajustarse a una determinada compañía antes de poder usarse.
- Requiere grandes esfuerzos iniciales e inversión para comenzar su uso.
- El manejo ad hoc de requerimientos.
- Comunicación imprecisa e incomprensible.
- Inconsistencia en requerimientos, diseño e implementación.
- Falta de pruebas.
- Estado de juicio subjetivo.
- Inhabilidad para manejar riesgos.
- Cambios sin control.
- Automatización de desarrollo ineficiente.

2.4.4 BUENAS PRÁCTICAS Y ADAPTABILIDAD DEL RUP, XP Y SCRUM

2.4.4.1 RUP

2.4.4.1.1 PROBLEMA ABORDADO Y SOLUCIÓN

Asegurar la alta calidad de la producción de software. Basada en los requerimientos de los usuarios finales [cumplimiento del cronograma y el presupuesto], provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que la adopte. Es guiado por casos de uso y centrado por la arquitectura, y utiliza el Lenguaje Unificado de Modelado [UML] como lenguaje de notación.

2.4.4.2 PROGRAMACIÓN EXTREMA

2.4.4.2.1 PROBLEMA ABORDADO Y SOLUCIÓN

- **Comunicación**. Contribuyen a maximizar la comunicación entre las personas, permitiendo de esa forma una mayor transferencia de conocimiento entre los desarrolladores y con el cliente, quien también es parte del equipo. Esto es logrado en la práctica gracias a la disposición física del lugar de trabajo. La idea es reunir a todas las personas en una misma oficina manteniendo una distribución denominada "cavernas y común".
- **Alto nivel de disciplina de las personas que participan en el proyecto**. XP permite mantener un mínimo nivel de documentación, lo cual a su vez se traduce en una gran velocidad en el desarrollo. Sin embargo, una desventaja que deviene de esta falta de documentación es la incapacidad de persistir la arquitectura y demás cuestiones de análisis, diseño e implementación, aún después de que el proyecto haya concluido.
- **Énfasis que pone en XP en las personas**. Se manifiesta en las diversas prácticas que indican que se deben dar más responsabilidades a los programadores para que estimen su trabajo, puedan entender el diseño de todo el código producido, y mantengan una metáfora mediante la cual se nombra las clases y métodos de forma consistente. La práctica denominada Semana de 40 horas indica la necesidad de mantener un horario fijo, sin horas extras ya que esto conlleva al desgaste del equipo y a la posible deserción de sus miembros. [Beck, 2000]. afirma que como máximo se podría llegar a trabajar durante una semana con horas extras, pero si pasando ese tiempo las cosas no han mejorado entonces se deberá hacer un análisis de las estimaciones de cada iteración para que estén acordes a la capacidad de desarrollo del equipo.

2.4.4.3 SCRUM

2.4.4.3.1 PROBLEMA ABORDADO Y SOLUCIÓN

- **Manejo de la impredictibilidad y el riesgo a niveles adaptables**. La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otras – como XP – para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un framework ágil de administración de proyectos que puede ser combinado con cualquiera de las metodologías mencionadas.
- **Al principio del proyecto se define el *Product Backlog*, que contiene todos los requerimientos funcionales y no funcionales que deberá satisfacer el sistema a construir.** Los mismos estarán especificados de acuerdo a las convenciones de la organización ya sea mediante: features, casos de uso, diagramas de flujo de datos, incidentes, tareas, etc. El Product Backlog será definido durante reuniones de planeamiento con los stakeholders. A partir de ahí se definirán las iteraciones, conocidas como Iteración [sprint] en la jerga de Scrum, en las que se irá evolucionando la aplicación evolutivamente. Cada Iteración [sprint] tendrá su propio Iteración [sprint] Backlog que será un subconjunto del Product Backlog con los requerimientos a ser construidos en el Iteración [sprint] correspondiente. La duración recomendada del Iteración [sprint] es de 1 mes.

[Schwaber, 2004] menciona que el Scrum, junto con Extreme Programming o XP, incluyen prácticas de mejora de la productividad que aumentan en gran magnitud la productividad de los equipos.

2.4.4.4 COMPATIBILIDAD DE LAS METODOLOGÍAS, XP, SCRUM Y RUP

Según [Núñez, 2010], no todas las metodologías ágiles contemplan todo el ciclo de vida tradicional. La metodología que contempla todas las etapas es la metodología Proceso Unificado Ágil [AUP], esto debido a que es una metodología que se basa en RUP. Pero a la vez, esta metodología sugiere que solo se utilice las etapas que sean necesarias para el proyecto.

2.4.4.4.1 CICLO DE VIDA DEL PROYECTO

	Principio del Proyecto			Espec. Requisitos			Análisis y Diseño			Codificación			Pruebas Unitarias			Pruebas de Integración			Pruebas Sistemas			Pruebas Acepción			Mantenimiento		
Metodología	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA	SG	PD	BA
Programación Extrema XP					X	X		X	X		X	X		X	X		X	X		X	X		X	X		X	X
Scrum				X	X	X	X			X			X			X						X	X		X		
Proceso Unificado Ágil [AUP]	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

SG: Soporte a la gestión.

PD: Se describe un proceso en el método que incluye esta etapa.

BA: Buenas prácticas, actividades y artefactos considerados en la etapa.

Tabla 2.3: Ciclo de vida tradicional de un proyecto software en las metodologías ágiles. [Núñez, 2010].

2.4.4.4.2 CALIDAD

Las metodologías ágiles están orientadas a la productividad, es por este motivo que en la comparativa de calidad solo algunas de las metodologías cumple con la gran parte de los parámetros de calidad.

Metodología	Fiabilidad	Usabilidad	Mantenibilidad
Programación Extrema [XP]	X	X	X
Scrum	X	X	
Dynamic Systems Development Method [DSDM]	X	X	X
Proceso Unificado Ágil [AUP]	X	X	X

Tabla 2.4: Comparativa de calidad en las metodologías ágiles. [Núñez, 2010].

NOTA:
Fiabilidad. Este parámetro viene determinado por los siguientes atributos: *Simplicidad,* de los diseños, en la implementación y en el desarrollo del software en general. *Trazabilidad,* entre los artefactos producidos en las distintas etapas del ciclo de vida del software.
Usabilidad. Este parámetro viene determinado por los siguientes atributos: *Claridad* y *Precisión* de la documentación. *Habilidades* que mejoren las pruebas del software.
Mantenibilidad. Este parámetro viene determinado por los siguientes atributos: *Modularidad*, esto ayuda a crear una documentación más fácil de entender. *Simplicidad,* si la metodología promueve que los sistemas desarrollados bajo su enfoque sean simples al momento de mantenerse.

2.4.4.4.3 HERRAMIENTAS

Existen muchas herramientas libres que ayudan en el proceso de las metodologías ágiles.

Metodología	Herramientas
Programación Extrema [XP]	• Herramientas para la realización de refactorización, IDEs como Eclipse y NetBeans. • Herramienta de integración continua: Cruise Control. • Herramientas de administración de proyectos y compilaciones automáticas: Maven y Ant. • Repositorio de códigos: CVS y Subversión. • Herramientas para pruebas unitarias: JUnit. • Herramientas para la medición de rendimiento de aplicaciones: JMeter.
Scrum	A pesar de no ser necesaria ninguna herramienta especial, están surgiendo aplicaciones Web que facilitan el seguimiento del proyecto y la generación de los distintos artefactos de la metodología, que frecuentemente se realizan con paquetes ofimáticas.
Proceso Unificado Ágil [AUP]	Para esta metodología son necesarias las herramientas mencionadas tanto para le metodología XP como para AMDD, además de estas herramientas se podría mencionar: • Herramientas de cobertura y evaluación de complejidad ciclomática: MAVEN.

Tabla 2.5: Herramientas de libre distribución para metodologías ágiles.

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1 LA INVESTIGACIÓN EN INGENIERÍA DEL SOFTWARE

[Marcos, 1998] responde a la pregunta ¿Cuál es el objetivo de estudio de la ingeniería de software?

Al dividir los problemas de la ingeniería de software según la naturaleza del conocimiento, una primera aproximación es:

a) Tipo A: La Investigación enfocada a la construcción de nuevos objetos [procesos, modelos, metodologías, técnicas, etc.].
b) Tipo B: Investigación enfocada al estudio de dichos objetos [métricas, optimización, etc.]. Su objeto de estudio no difiere de las ciencias tradicionales sino en que los objetos estudiados son artificiales en lugar de naturales.
c) Tipo C: Investigación enfocada a la implantación y uso de estos nuevos objetos.

El método de investigación para casos cuyo objetivo de estudio es del Tipo A, no es posible aplicar métodos empíricos, ya que el objeto de estudio aún no existe. Tienen un importante componente de creatividad, así como un fuerte componente social y cultural en cuanto a la adopción de nuevos paradigmas, la parte de trabajo en equipo, de los proceso de software, el trabajo colaborativo, etc. Por todo ello, y cada más se están empleando métodos de carácter cualitativo en la investigación de la ingeniería del software.

3.2 INVESTIGACIÓN EN ACCIÓN

[Cabrero, 2009] señala que existen diferentes métodos de investigación cualitativa, entre los que destaca el método Investigación-Acción. El término "Investigación-Acción" proviene de [Lewin, 1947] que lo utilizaba para describir una forma de investigación que podía enlazar el enfoque experimental de las ciencias sociales con programas de acción social que respondieran a los problemas sociales principales de entonces. Mediante la Investigación-Acción, [Lewin, 1947] argumentaba que se podían lograr en forma simultánea avances teóricos y cambios sociales. A pesar de que la primera propuesta de Investigación-Acción fue introducida en 1985 por [Wood-Harper, 1985], en los últimos años, ha obtenido una gran atención y aceptación por parte de la comunidad investigadora en Sistemas de Información [Avison et al., 1999, Seaman, 1999].

El objetivo final de este trabajo de investigación es el diseño de una Metodología de Desarrollo, aplicando los principios de la Ingeniería del Software. En este sentido, se empleó el método de Investigación-Acción como resultado de la colaboración del investigador y su aplicación en la fábrica de software. Los roles identificados en la Investigación-Acción [Wadsworth, 1998], son el investigador, el objeto investigado, el grupo crítico de referencia [GCR] y los beneficiarios.

La Tabla 3.1 muestra los roles asociados a nuestro caso concreto.

Nombre del Rol	**Descripción**	**Rol identificado en nuestro trabajo**
Investigador	El individuo o grupo que lleva a cabo de forma activa el proceso investigador	El tesista.
Objeto Investigado	El problema a resolver	Diseñar una Metodología de Desarrollo para una Fábrica de Software.
Grupo Crítico de Referencia [GCR]	Aquel para quien se investiga en el sentido de que tiene un problema que necesita ser resuelto y que también participa en el proceso de investigación [aunque menos activamente que	El conjunto de los ingenieros de software, así como los gestores de proyectos de tecnologías de la información de la fábrica de software.

Nombre del Rol	Descripción	Rol identificado en nuestro trabajo
	el investigador]	
Beneficiarios	Aquel para quien se investiga en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participa directamente en el proceso.	Cualquier desarrollador potencial de un sistema de información de la fábrica de software y los clientes externos.

Tabla 3.1: Roles identificados en la Investigación / Acción. Elaboración propia.

La Figura 3.1 muestra el flujo de propuesta, validación y refinamiento de propuestas, involucrando al investigador, el grupo crítico de referencia y los beneficiarios. El proceso consta de una fase de propuesta de técnicas, que será llevada a cabo por el investigador en conjunción con la Gerencia de Informática de fábrica de software. Posteriormente, se contrastan las técnicas a través de las validaciones realizadas por el investigador, con los datos de proyectos reales suministrados por la fábrica de software. Finalmente, se refinan las propuestas iniciales utilizando las mediciones obtenidas de las fases anteriores.

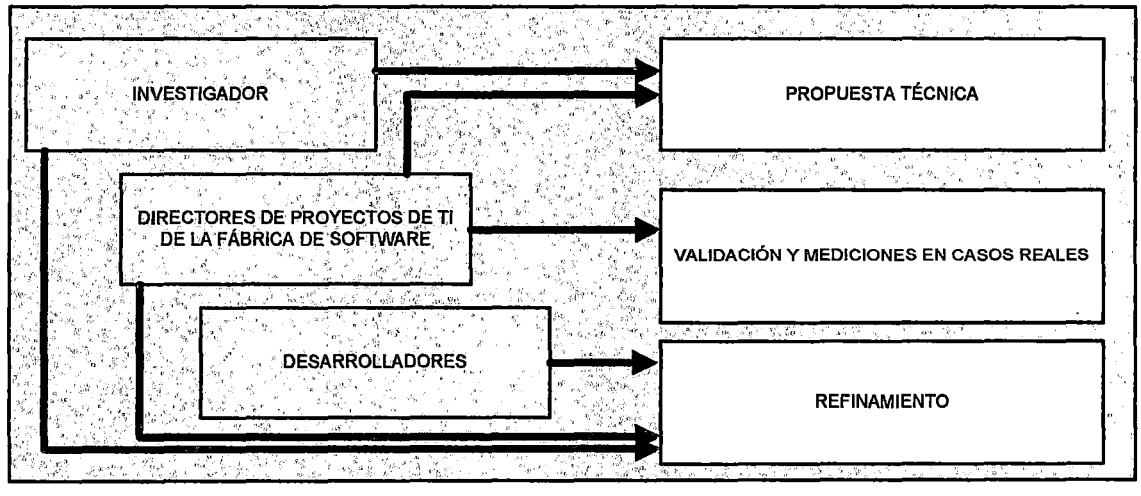


Figura 3.1: Método de investigación. Fases. Elaboración propia.

Según [Fernández, 2010], la investigación en acción [Avison et. al., 1999] es un método de investigación cualitativo utilizado para la validación de trabajos de investigación mediante su aplicación en proyectos reales,

reforzando la interacción entre los investigadores y los participantes de dichos proyectos reales. Éste método, basado por un carácter de validación práctica, es especialmente apropiado para la investigación en ingeniería y especialmente en ingeniería del software. En general, comprende los siguientes aspectos:

- Diagnóstico.
- Planificación en Acción.
- Ejecución en Acción.
- Evaluación.
- Especificación del Aprendizaje.

3.3 USO DE INDICADORES PARA MEDIR LA EFICACIA DE LA METODOLOGÍA

Una forma de determinar si la metodología que se adopta está cumpliendo con los objetivos esperados, es contar con un conjunto de indicadores que permitan a los equipos de trabajo evaluar las condiciones que se presentan. De esta manera, se puede determinar si es factible incorporar la metodología propuesta en la construcción de un producto software. Para llevar a cabo dicha acción, se debe evaluar la gestión total del proyecto, comparar los resultados obtenidos con criterios previamente establecidos y hacer un juicio de valor, tomando en cuenta la magnitud y dirección de la diferencia encontrada entre lo previsto y lo obtenido.

Un indicador aislado, obtenido una sola vez, puede ser de poca utilidad. En cambio, cuando se analizan sus resultados a través de variables de tiempo, persona y lugar; se observan las tendencias que el mismo puede mostrar con el transcurrir del tiempo y se combina con otros indicadores apropiados, se convierten en poderosas herramientas de gerencia, pues permiten mantener un diagnóstico permanentemente actualizado de la situación, tomar decisiones y verificar si éstas fueron o no acertadas.

3.3.1 CARACTERÍSTICAS DE UN BUEN INDICADOR

Un buen indicador se caracteriza por los siguientes atributos:

- Sirve a un propósito;
- Se ha diseñado teniendo en cuenta este propósito y las características de los usuarios;
- Guarda relación con un asunto de interés actual o futuro [es decir, es útil];
- Es costo-eficaz: logra el objetivo de su utilización con la mínima cantidad de recursos, utiliza recursos [datos, entre ellos] existentes o permite utilizar los datos nuevos que requiere para otros usos y usuarios;
- Es válido: es decir que mide lo que se pretende medir;
- Es objetivo: permite obtener el mismo resultado cuando la obtención del indicador es hecha por observadores distintos, en circunstancias análogas;
- Es sensible: es capaz de captar los cambios ocurridos en la situación objeto del indicador;
- Es específico: aplicable solo a la situación de que se trata;
- Es inequívoco en su significado;
- Se puede obtener sin dificultad;
- Es consistente en el transcurso del tiempo; se obtiene oportunamente;
- Es preciso;
- Es transparente [fácilmente entendido e interpretado por los usuarios]; es dado a conocer periódicamente a las partes interesadas.

3.4 POBLACIÓN Y MUESTRA

Se analizó el servicio que brinda la fábrica del software al cliente Entidad del Estado, Sistema Nacional de Pensiones [SNP]. La muestra fue de 19 tareas agrupadas en 5 iteraciones en la atención de un requerimiento de software.

3.4.1 OPERACIONALIZACIÓN DE VARIABLES

3.4.1.1 SISTEMA DE VARIABLES E INDICADORES

De las preguntas correspondientes al Problema General de Investigación y a los Objetivos Específicos de Investigación, anteriormente planteados, se obtiene las siguientes variables:

Variable Independiente [X]: "La Metodología de Desarrollo de Software", se mide mediante combinación de metodologías de desarrollo [RUP, XP y SCRUM].

Variable Dependiente [Y]: "Grado de cumplimiento de la tarea con las características especificadas", se mide mediante el siguiente indicador: Avance del producto software [Y1].

3.4.1.2 CLASIFICACIÓN Y DEFINICIÓN CONCEPTUAL Y OPERACIONAL DE LAS VARIABLES

3.4.1.2.1 VARIABLE INDEPENDIENTE - METODOLOGÍA DE DESARROLLO DE SOFTWARE

3.4.1.2.1.1 DEFINICIÓN CONCEPTUAL

Son los procesos de ingeniería compuesto por métodos, técnicas y herramientas necesarias para construir software de alta calidad.

3.4.1.2.1.2 DEFINICIÓN OPERACIONAL

X1: RUP. [Becerra, 2002] Es un proceso de ingeniería de software, bien definido y estructurado, a la vez que es un producto que provee un marco de proceso adaptable a las necesidades y características de cada proyecto específico. Según los creadores del RUP [Booch, Rumbaugh y Jacobson], ésta se basa en tres características fundamentales: primero, está dirigido por casos de uso; segundo, proceso centrado a la arquitectura y tercero, es iterativo e incremental.

X2: XP. [Becerra, 2002] La metodología XP enfatiza la satisfacción del cliente y promueve el trabajo en equipo. En XP, las actividades

improductivas han sido eliminadas para reducir costos y frustraciones. Esta metodología ha sido diseñada para solucionar el eterno problema de desarrollo de software por encargo, que es entregar el resultado que el cliente necesita a tiempo.

X3: SCRUM. "Es un proceso iterativo e incremental para desarrollar cualquier producto o administrar cualquier trabajo. Él produce un conjunto de funcionalidades potencialmente entregables al final de cada iteración."[23]

3.4.1.2.2 VARIABLE DEPENDIENTE - GRADO DE CUMPLIMIENTO DE LA TAREA CON LAS CARACTERÍSTICAS ESPECIFICADAS

3.4.1.2.2.1 DEFINICIÓN CONCEPTUAL

Procedimiento que permite validar el cumplimiento del desarrollo del producto software.

3.4.1.2.2.2 DEFINICIÓN OPERACIONAL

Y1: Avance del producto software. Mide el estado final del producto software aplicando la nueva metodología de desarrollo y mantenimiento.

[23] Ken Schwaber, "What is Scrum?" www.controlchaos.com.

CAPÍTULO IV

METODOLOGÍA DE DESARROLLO PROPUESTA

Las metodologías seleccionadas son SCRUM, XP y RUP. De estas metodologías se ha analizado, tanto las mejores soluciones como su planteamiento metodológico, donde en base a este análisis, se ha llegado a la conclusión, de utilizar ciertas actividades y prácticas de cada una de estas metodologías y así obtener un marco ágil que se ajuste a lo requerido.

Las actividades y prácticas seleccionadas, tanto de las metodologías Scrum como RUP, formarán la estructura del marco ágil de trabajo. Esto debido, al enfoque que tienen estas metodologías, consiguiendo de esta manera, los lineamientos para la gestión [planteada por SCRUM] y las formalidades del Proceso Unificado [planteada por RUP] en la ejecución del ciclo de vida del software, con lo cual, se dejará a la metodología XP enfocada directamente en el plano de desarrollo.

Siguiendo esta idea, se ha optado por utilizar ciertas fases y actividades que plantea RUP, las cuales serán gestionadas por los lineamientos de SCRUM. Consiguiendo de esta manera también incluir prácticas de XP en este marco ágil de trabajo.

4.1 ASPECTOS A CONSIDERAR EN LA METODOLOGÍA

Se consideran los siguientes aspectos:

- Objetivos de la metodología.
- Restricciones

- Roles.
- Fases e hitos.
- Disciplinas dentro de las fases.
- Disciplinas de soporte.
- Artefactos.

4.1.1 OBJETIVOS DE LA METODOLOGÍA DE DESARROLLO PROPUESTA

Como principales objetivos se consideró lo siguiente:

- La metodología debe permitir desarrollar sistemas de software con la calidad requerida y a satisfacción del usuario.
- La metodología debe permitir desarrollar gran variedad de sistemas de software.
- La metodología debe soportar todas las etapas de desarrollo de software.
- La metodología debe permitir superar o mitigar las limitaciones o desventajas de las metodologías de desarrollo de software tradicional, en lo referente a facilitar el desarrollo y mantenimiento de los sistemas software.

4.1.2 RESTRICCIONES DE LA METODOLOGÍA

La metodología propuesta considera las siguientes restricciones:

- Cantidad de personas involucradas. [Cockburn, 2000] muestra que este aspecto es esencial a la hora de elegir una metodología [ver Figura 4.1]. La metodología propuesta en particular está enfocada a pequeños grupos de desarrollo, de no más de 15 personas involucradas en el proyecto, las cuales deben compartir un ambiente de trabajo común, con mínima separación física. Cabe mencionar que en caso de contar con una mayor cantidad de recursos se deberán dividir en subgrupos del tamaño antes indicado, trabajando en forma paralela.

- Otra variable a tener en cuenta, es el grado de criticidad de la aplicación. A medida que aumenta el perjuicio para el negocio ante una posible falla en el sistema, se deberá utilizar una metodología más pesada, con procesos de revisión bien pautados, de forma de asegurarse la calidad del producto. No es lo mismo construir una aplicación empaquetada para bajar archivos de internet automáticamente, que el software utilizado en un equipo de resonancia magnética. En el primer caso, una pérdida solo causa la posible pérdida de información para un usuario ocasional. En el segundo caso, una falla podría resultar en anomalías en los resultados obtenidos con la posibilidad de realizar diagnósticos erróneos. Aquí se pone de manifiesto, la otra magnitud provista en la Figura 4.1; en la cual, se observa que a medida que aumenta la criticidad, desde la pérdida de *Confort* hasta la pérdida de una Vida, se debe agregar más rigor a la metodología para estar seguros de entregar un software lo más confiable posible. Es así que se postula la metodología propuesta como una metodología para ser utilizada en proyectos de características C3, C10, D3, que tienen como máximo 20 personas en el equipo de desarrollo y que presentan un grado de criticidad bajo de pérdida de *Confort* o pérdida Económica Discrecional. En la fábrica de software se tiene una distribución por cada sistema de información un promedio de 30 personas y a su vez esta subdivida por línea de producción las cuales están conformadas por 3 personas tal como se muestra en la Figura 4.2.

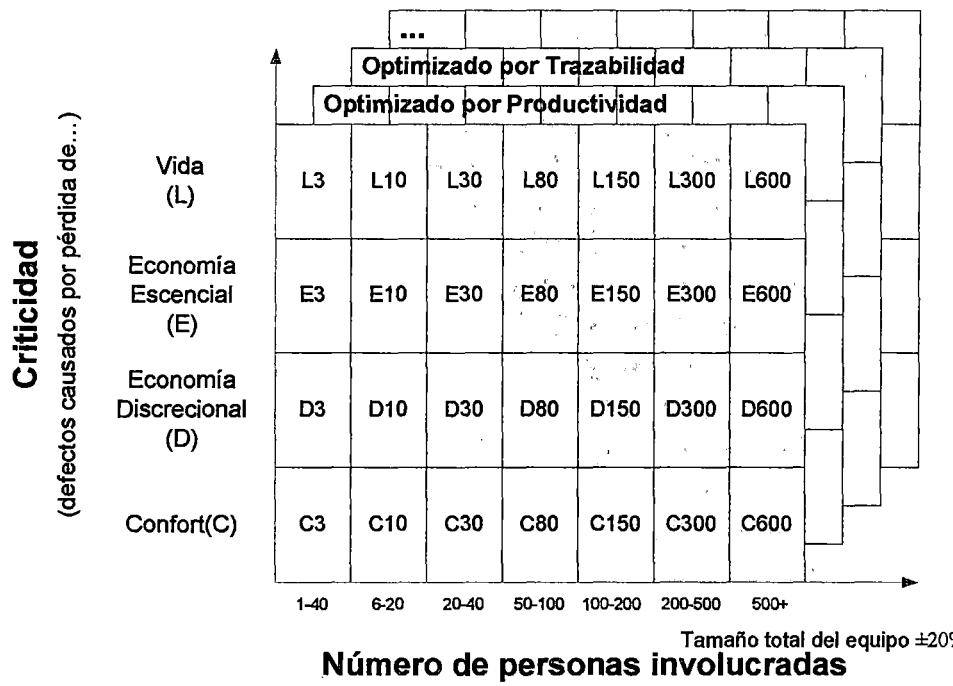


Figura 4.1: Relación entre cantidad de personas, criticidad, y tamaño de la metodología. Tomada de [Cockburn, 2001].

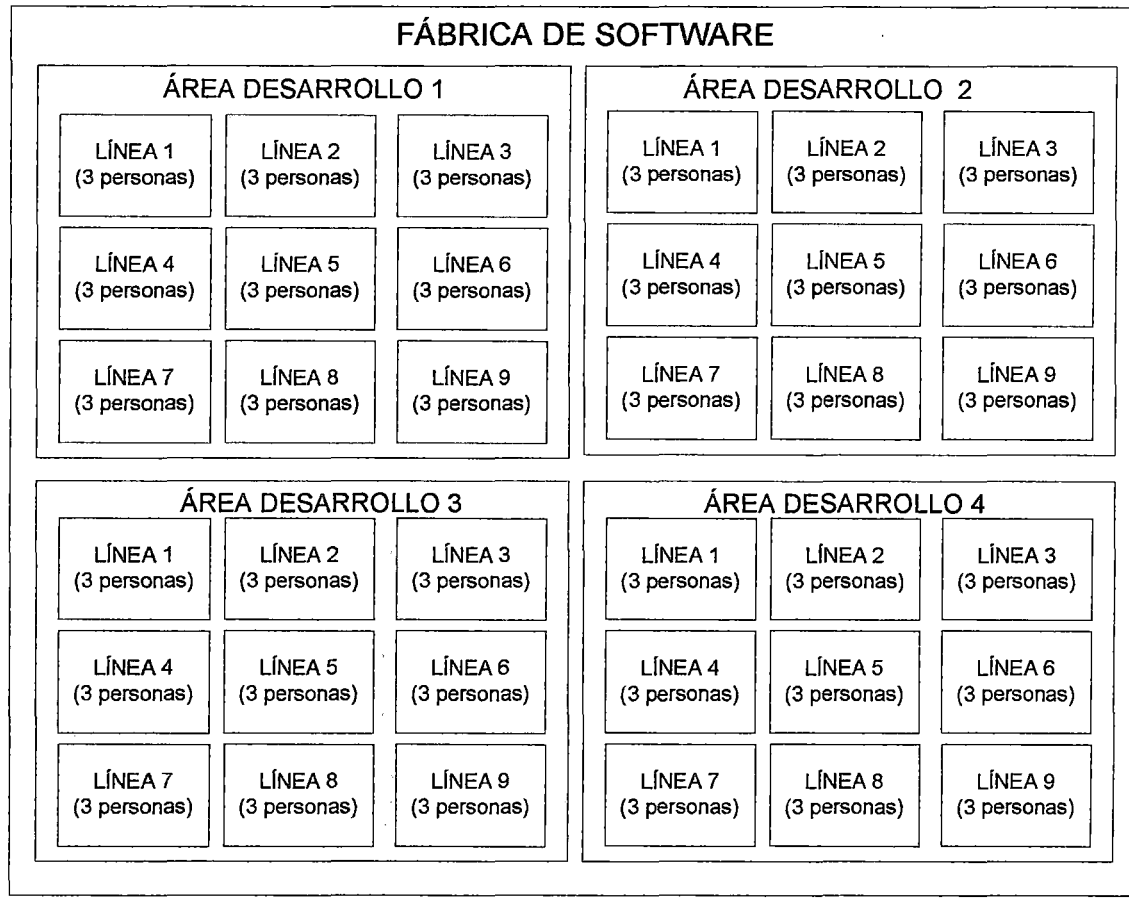


Figura 4.2: Distribución de las líneas de producción en la fábrica de software.

- La principal necesidad que cubre la metodología, se enfoca en identificar factores [principios ágiles] para mejorar las aplicaciones. No se profundiza el análisis de procesos de desarrollo.
- La metodología tiene como objetivo manejar la impredictibilidad y adaptabilidad de los proyectos [SCRUM], mejorar la captura de requisitos y desarrollo [XP] y calidad de software [RUP].

4.1.3 ROLES

Los roles definen las actividades realizadas y los artefactos a desarrollar por las personas o grupos de personas del proyecto, involucran a otras personas externas que tenga que ver con el proyecto. Una persona puede cubrir más de un rol. Las asignaciones serán llevadas a cabo por el

Líder, el cual en base a las aptitudes de los recursos que dispone repartirá las tareas a ser realizadas en cada iteración. [Cockburn 2000] menciona lo siguiente "Es malo para el proyecto cuando los individuos no tienen las características necesarias para su puesto de trabajo [por ejemplo un gerente de proyecto que no puede tomar decisiones o de un mentor que no le gusta comunicarse". La asignación de los roles están relacionadas con las habilidades inherentes de la persona que participa del proyecto.

Para la metodología propuesta se definió los siguientes roles:

4.1.3.1 PATROCINANTE

Actividades: tiene a su cargo el soporte gerencial del proyecto; es el encargado de proveer, comunicar y mantener actualizada la Visión del proyecto; provee el presupuesto para la viabilidad económica del desarrollo; es responsable por la consecución del proyecto del lado del cliente.

Importancia del rol: es esencial para el éxito del mismo, ya que un software que no tiene aceptación dentro de la organización que lo financia jamás llegará a ser construido en tiempo, forma y con consentimiento de los usuarios, no siendo utilizado eventualmente si se concreta el proyecto.

4.1.3.2 LÍDER DE PROYECTO

Actividades: tiene a su cargo la planificación del proyecto, a lo largo de todo el ciclo de vida, incluida la planificación en detalle de cada iteración; asigna recursos y delega responsabilidades en los mismos; fomenta la cohesión del grupo y lleva a cabo actividades destinadas a eliminar fricciones; organiza las reuniones a ser realizadas; monitorea el progreso del proyecto y establece estrategias para mitigar los riesgos que se puedan presentar.

Importancia del rol: el Líder de Proyecto representa la cara visible del equipo de desarrollo, es el nexo existente entre la gerencia y el equipo de desarrollo como se observa en la Figura 4.3.

4.1.3.3 EXPERTO EN EL DOMINIO

Actividades: tiene a su cargo brindar su conocimiento del negocio contribuyendo al modelado del sistema que llevan a cabo los Analistas durante la disciplina de Requerimientos-Análisis; participará junto con los analistas de pruebas en la definición del contenido de las pruebas funcionales a ser realizadas; será el responsable de la aprobación de las pruebas de aceptación por cada release entregado.

Importancia del rol: el Experto en el Dominio permite al Equipo de Desarrollo aprender sobre el negocio para el cual está siendo construida la aplicación; son encargados de resolver cualquier cuestión relacionada con la funcionalidad de la aplicación junto con los Analistas.

Destrezas: el Experto en el Dominio deberá conocer en detalle el negocio para prestar respuesta a cualquier duda que pueda surgir del mismo. En general será un miembro de la empresa Cliente.

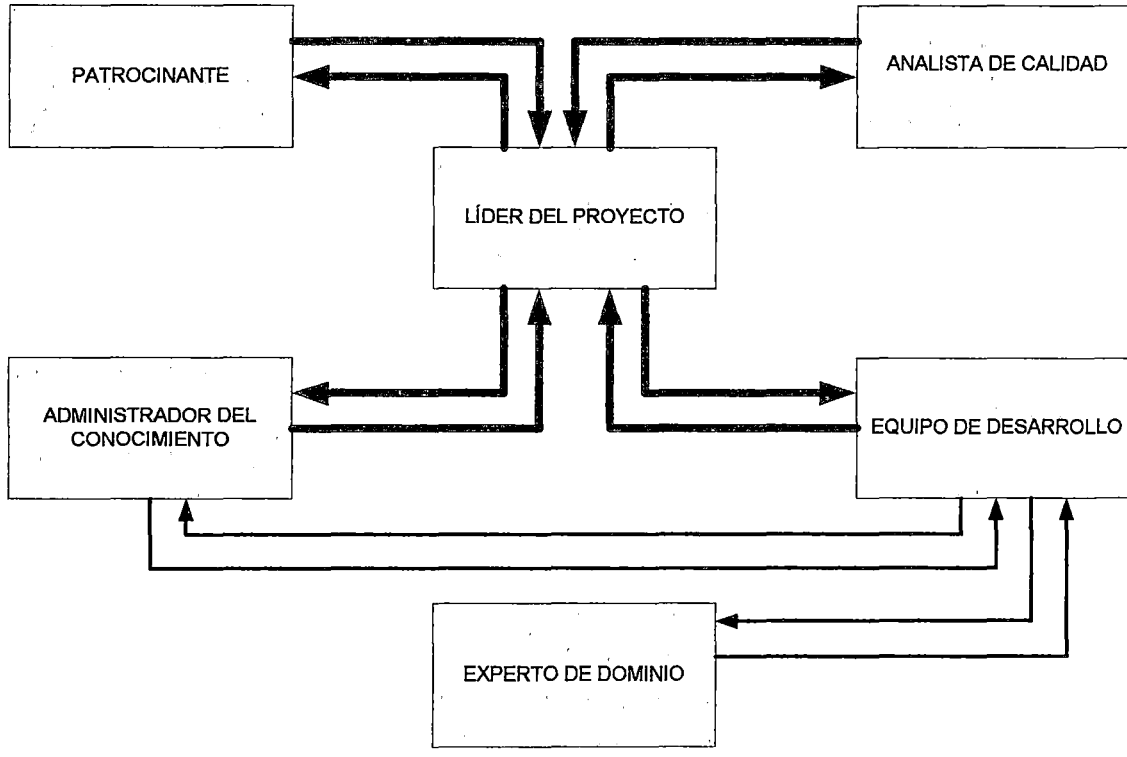


Figura 4.3: Flujos de comunicación durante el proyecto entre los roles propuestos.

4.1.3.4 COORDINADOR

Actividades: tiene a su cargo la supervisión del proceso, y cualquier actividad orientada al mejoramiento del mismo. Durante las primeras etapas de utilización de la metodología propuesta supervisará la implementación del proceso.

Importancia del rol: en las metodologías ágiles este rol permite reforzar la adherencia al proceso en aquellos momentos en que se el tiempo apremia y se suele caer en el modelo Codificar y Probar.

Destrezas: el Experto en el Dominio deberá conocer en detalle el negocio para prestar respuesta a cualquier duda que pueda surgir del mismo.

4.1.3.5 ANALISTA

Actividades: tiene a su cargo el relevamiento, mediante el cual se obtienen los requerimientos de la aplicación a ser construidos en cada iteración; realiza la especificación de los requerimientos; prepara el documento de Visión.

Importancia del rol: el aprendizaje del dominio de la aplicación y de los requerimientos que deberá tener la misma son claves para el éxito del proyecto y la aceptación del mismo por parte del usuario.

Destrezas: el Analista deberá tener amplio conocimiento de técnicas de relevamiento, así como aptitudes sociales que le permitan vencer el "Síndrome del Usuario y el Desarrollador" [Leffingwell, 2001].

4.1.3.6 ARQUITECTO

Actividades: tiene a su cargo la definición de la arquitectura que guiará el desarrollo, y de la continua refinación de la misma en cada iteración; deberá construir cualquier prototipo necesario para probar aspectos riesgosos desde el punto de vista técnico en el proyecto; definirá los lineamientos generales del diseño y la implementación.

Importancia del rol: la arquitectura es imprescindible en los proyectos de software actuales en donde cada vez existe mayor complejidad; el

arquitecto puede ser considerado como el Experto en la parte técnica del desarrollo y debe mantener a todo el equipo en conocimiento de los lineamientos fundamentales de la construcción.

Destrezas: el Arquitecto deberá tener una buena formación técnica, contar con experiencia en las herramientas y técnicas utilizadas; aptitudes comunicacionales son deseadas para que la arquitectura sea comunicada a todos los miembros del equipo; también deberá ser perseverante en conseguir los hitos técnicos planteados mediante entregables para asegurar el progreso de la construcción.

4.1.3.7 PROGRAMADOR O DESARROLLADOR

Actividades: tiene a su cargo la codificación de los componentes a desarrollar en la iteración; debe crear y ejecutar las pruebas unitarios realizados sobre el código desarrollado; es responsable de las clases que ha desarrollado debiendo documentarlas, actualizarlas ante cambios y mantenerlas bajo el control de configuración de las mismas mediante la herramienta de Gestión de Configuración del Software utilizada.

Importancia del rol: el Programador es la persona que tiene los materiales y lleva a cabo la implementación de los casos de uso en el lenguaje de programación elegido; como en general, es el paradigma de objetos [Como se observa por estudios diversos de Gartner, Forrester, entre otros]. Las dos plataformas que eventualmente comprenderán el mayor porcentaje de los futuros desarrollos, Java y .NET, están construidos sobre lenguajes orientados a objetos los cuales basan su comportamiento en clases, atributos y métodos.], el que está imponiéndose en la industria de IS/IT, el Programador definirá las clases y métodos que realicen los correspondientes casos de uso.

Destrezas: el Analista deberá tener amplio conocimiento de las herramientas de desarrollo, del lenguaje de programación, de los aspectos técnicos involucrados.

4.1.3.8 ANALISTA DE PRUEBAS

Actividades: tiene a su cargo la generación de pruebas funcionales a partir de los requerimientos extraídos por los Analistas.

Importancia del rol: la importancia del Analista de Pruebas radica en la necesidad de construir un software de calidad que cumpla con los requerimientos del usuario; mediante la utilización de un proceso y el armado de un grupo cohesivo de desarrollo, se tienen prácticas para garantizar la calidad en el producto desde el punto de vista técnico. Sin embargo, para asegurarnos de que la aplicación satisface las necesidades del usuario se debe realizar todo tipo de pruebas de carácter funcional. Es ahí en que entra en juego el Analista de Pruebas, quien crea, ejecuta, analiza y mantiene el conjunto de pruebas automatizadas y manuales que son utilizados.

Destrezas: el Analista de Pruebas deberá tener amplio conocimiento de técnicas de pruebas, deberá conocer a fondo la aplicación que probará. Asimismo, deberá tener conocimientos de programación para trabajar con las pruebas automatizadas.

4.1.3.9 ADMINISTRADOR DEL CONOCIMIENTO

Actividades: tiene a su cargo la captura, refinamiento, empaquetamiento, y transferencia del conocimiento, ya sea tácito o explícito, en la organización. En particular, la metodología propuesta recomienda que este rol sea llevado a cabo por una persona del equipo de desarrollo con dedicación medio tiempo – esto puede depender del tamaño de la organización y de otros factores.

Importancia del rol: el Administrador del Conocimiento es uno de los pilares sobre los que se establece Metodología propuesta. Su importancia consiste en la capacidad del equipo de desarrollo de aprender de la experiencia que éste y que otros equipos dentro de la organización generan a diario durante el transcurso de los proyectos. Mediante esta disciplina se logra el reuso de dicho conocimiento.

Destrezas: el Administrador del Conocimiento debe poseer aptitudes en comunicación para poder capturar el conocimiento de aquellas personas que lo generan.

4.1.3.10 ANALISTA DE CALIDAD

Actividades: Tiene como objetivo asegurar el cumplimiento de estándares y procedimientos relacionados al desarrollo de sistemas, proyectos y/o servicios de negocios especializados.

Importancia del rol:

- Es importante porque establece un plan para la gestión de la calidad que permite lograr y mantener la calidad y la seguridad de la información en el proceso de software.
- Analiza y optimiza los controles de calidad del desarrollo y mantenimiento de los sistemas asociados al servicio.
- Identifica oportunidades de mejora en la calidad del servicio.
- Propone puntos de control en el desarrollo y mantenimiento de los sistemas del servicio, que ayuden a optimizar la calidad y seguridad de los mismos.
- Establece las normas y estándares de calidad pertinentes con el fin de garantizar la eficacia del mantenimiento y desarrollo de sistemas.
- Realiza auditorías de calidad a los sistemas.
- Realiza seguimiento al cumplimiento del cronograma del plan de calidad del servicio.
- Propone indicadores que midan la calidad del servicio.
- Propone políticas de calidad para el servicio.
- Actualiza y ejecuta el plan de mejora continua del servicio.
- Actualiza y ejecuta la metodología de aseguramiento de la calidad de software del servicio.

4.1.4 FASES E HITOS

La metodología propuesta abarca un conjunto de fases, las cuales comprenden dos perspectivas de desarrollo.

- La primera perspectiva. Ocurre en las primeras iteraciones del proyecto, cuando:
  - Se analiza *la factibilidad de la ejecución* del mismo y
  - Se termina con *la obtención de la arquitectura* con los riesgos más críticos mitigados.
- La segunda perspectiva. Ocurre en fases iterativas de construcción en donde el énfasis esta en el código fuente generado.

Esto significa que el grado de creatividad requerido en el proceso disminuye con el tiempo, tornándose en un ciclo más predecible. En relación a la definición de fases e hitos de la metodología propuesta se han tomado dos fuentes importantes [Boehm, 1995] y [RUP, 2002].

4.1.4.1 CONCEPCIÓN

Que consiste en:

- La definición de las características que tendrá la aplicación, el alcance de la misma.
- La identificación de los stakeholders.
- La personalización del proceso a ser usado y
- Llevar a cabo la planificación general del proyecto.

Esta fase provee el fundamento en que todo el trabajo posterior se basa. Es crítico en esta fase llevar a cabo los primeros relevamientos que se realizan con el cliente de forma de adquirir conocimiento del dominio y analizar si los costos del proyecto estarán justificados o bien conviene comprar algún software empaquetado o cancelar la ejecución.

La fase de Concepción finaliza con un hito mayor denominado Objetivos del Ciclo de Vida, en el que se evalúa lo realizado contra las expectativas del Cliente y del Equipo de Desarrollo.

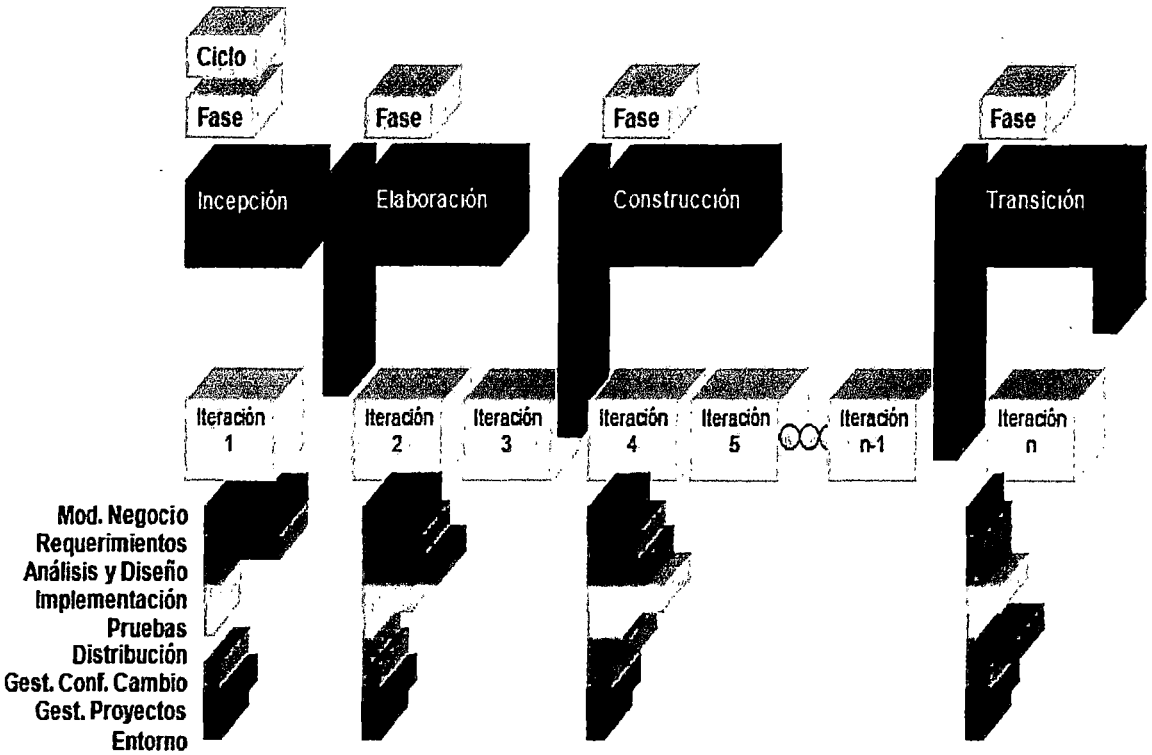


Figura 4.4: Fases que componen la Metodología propuesta

4.1.4.2 ELABORACIÓN

Se refiere a:

- La exploración de los requerimientos más críticos [funcionales y no funcionales] que involucra el proyecto, así como las decisiones técnicas más importantes que quedarán plasmadas en el Documento de Arquitectura.
- El objetivo principal consiste en asegurar la factibilidad técnica respecto a la realización del proyecto.
- Es a partir de la próxima fase cuando se comienza con la construcción a gran escala del software, en donde se comprometen la totalidad de los recursos necesarios para que el desarrollo se complete en las iteraciones planificadas.
- Asimismo, se podrán incorporar recursos en forma limitada tratando de no obstaculizar el normal transcurso del proyecto debido a la capacitación que estos últimos requerirán.

La fase de Elaboración finaliza con un hito mayor denominado Arquitectura del Ciclo de Vida en el que se evalúa lo realizado contra las expectativas del Cliente y del Equipo de Desarrollo.

Una vez que se pasa a las etapas de producción se tiene dos fases concatenadas que se realizan en forma repetitiva por cada *release* de la aplicación. Estas fases son Construcción y Transición.

4.1.4.3 CONSTRUCCIÓN

- Se terminan de especificar los casos de uso correspondientes a la iteración, se diseñan los mismos bajo la arquitectura candidata presentada.
- Se codifican todos los componentes definidos por los casos de uso.
- Ejecutándose las pruebas correspondiente y la integración. Cuando se quiera pasar un cierto conjunto de componentes al entorno productivo se tendrá una fase de transición.

4.1.4.4 TRANSICIÓN

- En la cual se llevarán a cabo las actividades de despliegue necesarias [ver Figura 4.4 con Iteración con Release].
- La fase de Transición finaliza con un hito mayor denominado Release del Producto en el que se evalúa lo realizado contra las expectativas del Cliente y del Equipo de Desarrollo.
- Cabe mencionar que dentro de las iteraciones de construcción se llevan a cabo actividades relacionadas con Requerimientos, Diseño, Pruebas, etc., ya que se trata de un proceso iterativo e incremental es que se va llevando a cabo tareas en paralelo y la aplicación va evolucionando hasta cumplir los casos de uso definidos.

4.1.5 DISCIPLINAS DENTRO DE LAS FASES

Para planificar un proyecto y permitir tener visibilidad sobre el progreso y el grado de avance del mismo, la metodología propuesta propone dividir el proceso en disciplinas que conforman agrupaciones de actividades en las cuales se produce un conjunto particular de artefactos. Las disciplinas cubiertas son las que se muestran en la Figura 4.5 y se irán realizando dentro de cada fase con diversos grados de paralelismo entre ellas.

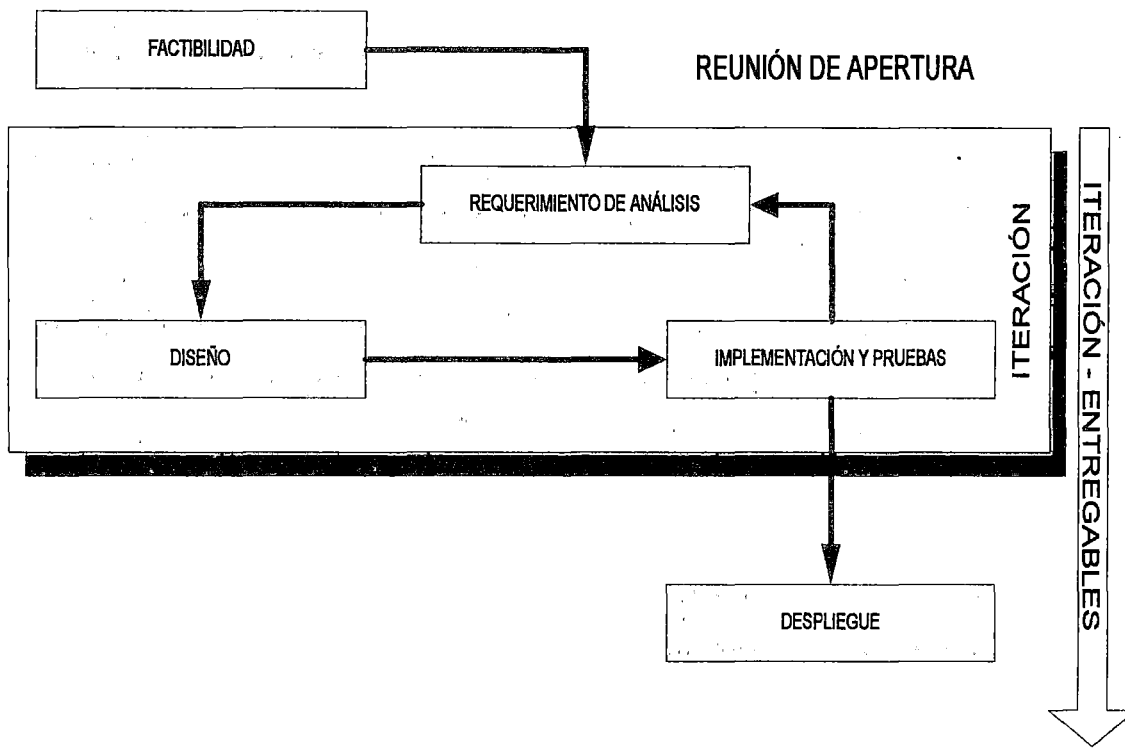


Figura 4.5: Disciplinas que componen Metodología propuesta.

4.1.5.1 FACTIBILIDAD

- Durante la disciplina de Factibilidad se produce el primer contacto entre el equipo de desarrollo y el cliente.
- Esta disciplina permite al equipo de desarrollo adquirir todo el conocimiento posible acerca del dominio, las necesidades de los usuarios, los objetivos y expectativas que debe cumplir el software que está siendo construido.

- Mediante las actividades incluidas en esta disciplina, se puede responder las siguientes preguntas:
    - ¿Es conveniente construir la aplicación?
    - ¿Es conveniente para el cliente el desarrollar un sistema con todas las complejidades inherentes para satisfacer las necesidades del usuario?
    - ¿Cuáles son las posibles soluciones que se pueden plantear?
- A medida que se ejecuta esta disciplina, se va realizando el:
    - **Modelado del Dominio**. Tendiente a entender las operaciones que forman parte del dominio del problema, el dominio manejado por el cliente. En forma conjunta, los analistas comenzarán a entender el problema planteado comenzando a bosquejar cuáles son las características que tendrá el sistema.
    - **Arquitectura Preliminar**. El arquitecto comenzará a realizar modelos y diagramas planteando una arquitectura preliminar que pueda servir de solución al problema en cuestión.
- Durante esta disciplina el énfasis está puesto en la exploración tanto del problema como de la solución.
- Es esencial el aprendizaje del dominio del cliente, de cómo el sistema puede solucionar los problemas que se presentan, de cómo esta solución puede ser realizada con las herramientas disponibles, de las restricciones impuestas sobre el sistema.
- Como se observa en la Figura 4.6, el entendimiento del negocio permite al equipo de desarrollo conocer explorar *soluciones adecuadas a las necesidades de los usuarios*; este conocimiento puede ser plasmado en un documento de ***Modelo del Dominio*** que debe ser construido conjuntamente con los usuarios y validados por estos.

- Mientras se va desarrollando el ***Modelo del Dominio***, se van explorando las distintas posibilidades en relación al dominio de la solución.
- La **solución técnica propuesta**
    - Deberá ajustarse a los estándares que posea el cliente, y el equipo de desarrollo
    - Deberá asegurarse de la factibilidad de su implementación.
- Para este punto se puede tener un documento de ***Descripción de Arquitectura*** que detalle las decisiones técnicas tomadas a lo largo de la Factibilidad.

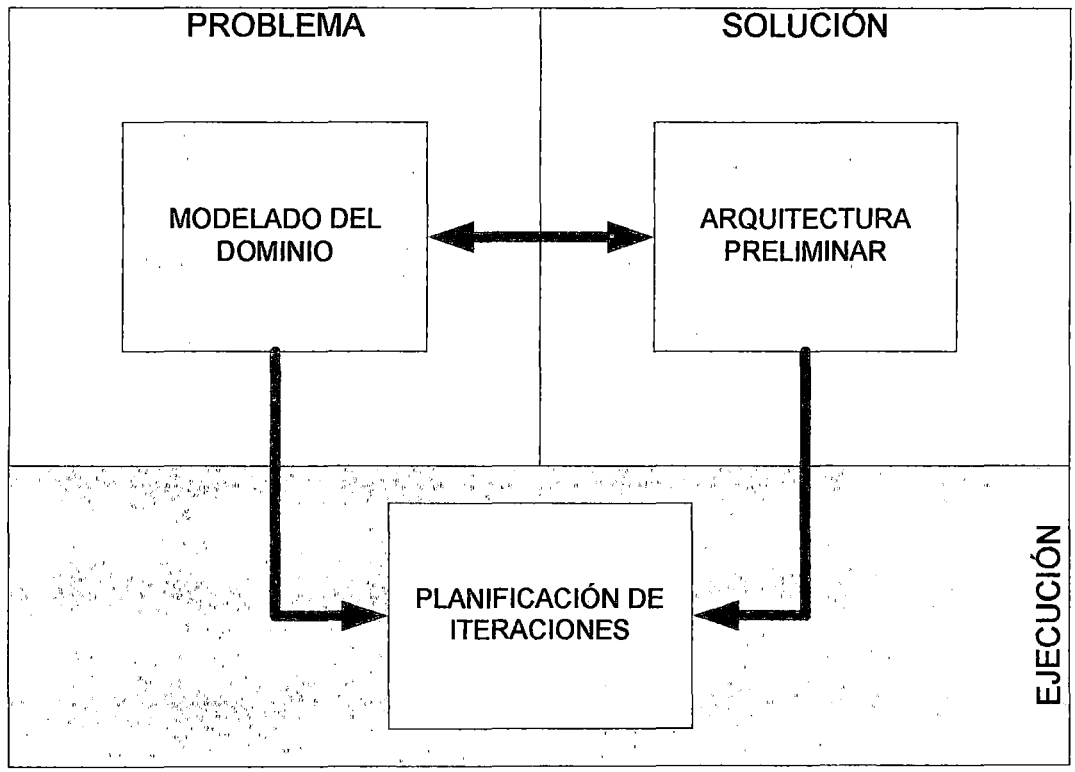


Figura 4.6: Factibilidad en detalle.

- Como se observa en el detalle, una vez que se finaliza la exploración y se han creado los documentos que comuniquen el resultado de la misma al cliente, se deberá *tener concurrencia*

de este y de los demás stakeholders en relación al compromiso para que el proyecto sea continuado o cancelado.

- Es de gran importancia el contar con el soporte ejecutivo necesario para proseguir esta disciplina, ya que según lo reportado por [Standish, 1994] la segunda razón por la cual un proyecto puede alcanzar el éxito es Soporte de la Gerencia Ejecutiva.
- El momento para garantizar este soporte es posteriormente a la exploración de los dominios ya mencionados, justo antes de la **Planificación de las Iteraciones** en que será construido el sistema.
- La Factibilidad finaliza con la **planificación del contenido funcional** que será desarrollado en cada una de las iteraciones en que se construirá la aplicación. En dicho plan, se determinarán cuáles serán las iteraciones con *release*, las cuales servirán como hitos del progreso del proyecto tanto para el cliente como para el equipo de desarrollo.
- Dicha planificación reviste un carácter preliminar ya que los requerimientos pueden presentar cambios que alteren el contenido de las funcionalidades a ser implementadas en cada iteración. Está planificación contendrá los elementos más importantes y será plasmada en un artefacto denominado **Plan de Proyecto**.

4.1.5.2 REQUERIMIENTOS-ANÁLISIS

Una vez completadas las actividades dentro de la disciplina de Factibilidad, comienza el desarrollo netamente iterativo. Las iteraciones se suceden mediante la realización de actividades relacionadas con las subsiguientes disciplinas. A continuación, se explica en detalle la primera disciplina con que comienzan las iteraciones de la metodología propuesta.

- En la disciplina de Requerimientos-Análisis se realizan actividades tendientes a capturar las necesidades de los stakeholders, transformar las mismas en características de alto nivel y especificarlas posteriormente en casos de uso. Como se observa en la Figura 4.7, se deberá especificar el espectro funcional el cual servirá para que los desarrolladores diseñen e implementen los casos de uso y también el **espectro no funcional** que posteriormente utilizará el arquitecto para: construir la arquitectura de la aplicación y el prototipo.

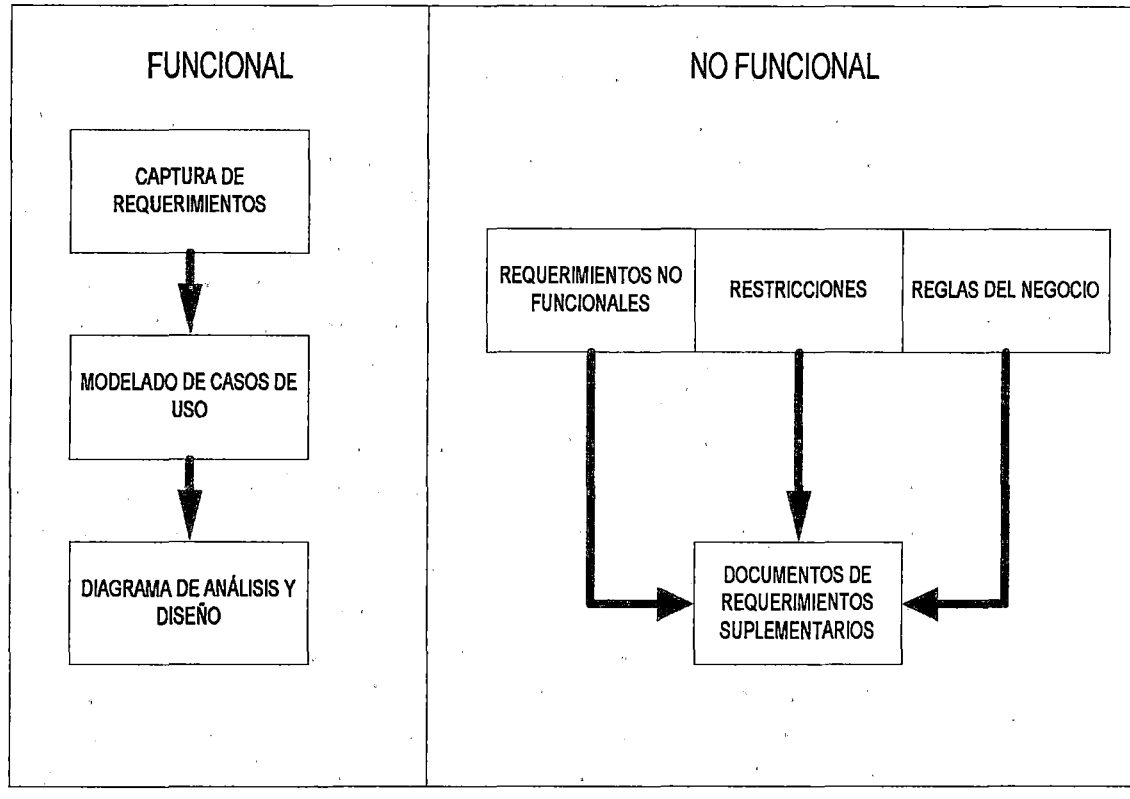


Figura 4.7: Requerimientos-Análisis en detalle.

- Dadas las características de la metodología propuesta tendientes a seguir un proceso ágil que puede administrar requerimientos altamente volátiles se sugiere utilizar el patrón denominado **Orientación al Cliente**, el cual consiste en

Maximizar la Comunicación directa con las personas que terminarán utilizando el sistema para asegurarnos de estar construyendo lo que estas necesitan. En otras palabras el proceso será más efectivo si se tiene a un Cliente al menos trabajando como parte integral del equipo de desarrollo [Beck, 2000]. Su principal función será la de instruir y transferir su conocimiento al resto del equipo a medida que transcurre el proyecto. Muchas veces esto resulta impráctico en cuyo caso se deberá asegurar el involucramiento de los usuarios y una frecuente validación por los mismos de los artefactos construidos en cada fase.

- Para llevar a cabo esta transferencia de conocimiento desde los usuarios y stakeholders que forman parte del proyecto a los analistas del equipo de desarrollo, se utilizan diversas técnicas las cuales han sido tratadas extensivamente en la bibliografía del tema; se puede ver en [Leffingwell, 2001][Kulak, 2003] una descripción de las más relevantes. De estas **técnicas** que se observan, la metodología propuesta recomienda utilizar aquellas que no demanden gran cantidad de recursos o una significativa infraestructura. Una de las técnicas más utilizadas en los procesos ágiles son:
    - Las entrevistas. Las cuales tendrán un carácter más informal ya que se podrán dar en cualquier momento que el analista deba aclarar cuestiones respecto a los requerimientos, detallar aspectos de un caso de uso, etc.
    - Sesiones de Brainstorming. En caso que se necesite hacer participar a un número importante de stakeholders en el relevamiento, se recomiendan sesiones de brainstorming dirigidas por el Líder de Proyecto a las cuales asisten los analistas documentando todo las decisiones tomadas.

- Posteriormente a la **Captura de Requerimientos** se realiza la primera aproximación de los mismos en una forma estructurada que consiste en los casos de uso, propuestos por Ivar Jacobson a principios de los noventa. Los mismos se han transformado en un estándar en la industria de la informática y han trascendido más allá de la comunidad de objetos de donde surgieron para ser utilizados ampliamente en proyectos de todo tipo en el marco de IS/IT. La principal ventaja de los mismos es que son escritos en el lenguaje del cliente y no demandan tener conocimientos técnicos para ser entendidos. Son esenciales en los procesos ágiles en que se intenta que exista comunicación constante y fluida entre los miembros del equipo y el Cliente.
- Para escribir los casos de uso se recomienda utilizar un formato predefinido. Esto permitirá que los analistas del equipo de desarrollo sincronicen el nivel de ambigüedad de la narración, los atributos a ser escritos y las convenciones que se utilizarán. Más material respecto al nivel de detalle con el que se puede realizar un caso de uso y todo aquello relacionado con su construcción ha sido estudiado por [Cockburn, 2000a].

4.1.5.3 DISEÑO

La disciplina de Diseño consiste:

- En plasmar el problema planteado [que se halla en forma de casos de uso o requerimientos contenidos implícitamente por Analistas o Clientes] en **el dominio de la solución**.
- Asimismo, también se realiza la **definición de la arquitectura** siendo validada mediante algún prototipo [conocidos *como Proof Of Concept*].

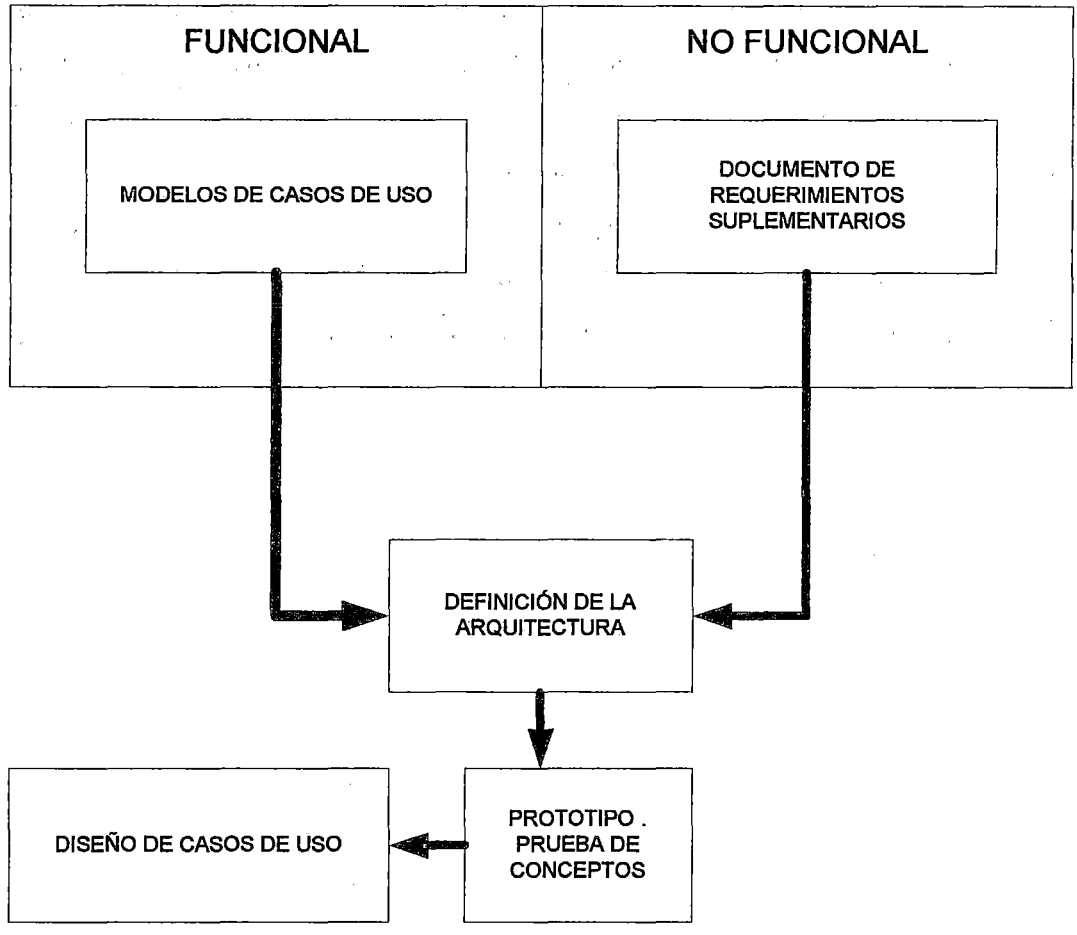


Figura 4.8: Diseño en detalle

- Durante las primeras iteraciones cabe destacar la necesidad de **diseñar la infraestructura** sobre la cual será construido el software. Para esto el arquitecto tomará del conjunto de casos de uso relevados hasta el momento aquellos que considere más importantes desde un punto de vista técnico.
- Asimismo, analizará los **requerimientos suplementarios** que deberán ser satisfechos por la arquitectura propuesta. Con esto construirá un **modelo de arquitectura** documentado que especificará los elementos a partir del cual será construido el sistema, las interacciones entre dichos elementos, los patrones que se usaron para su composición, y las restricciones sobre dichos patrones.

- Finalmente, para probar la factibilidad de la arquitectura el Arquitecto construirá un **prototipo ejecutable** utilizado para verificar la conformidad a los requerimientos de la aplicación.
- Dentro de Metodología propuesta, el Diseño es realizado tomando como entradas los casos de uso escritos en la disciplina anterior y detallando los mismos en un esquema de clases, atributos y métodos del lenguaje elegido.
- Los miembros del equipo adaptan su modelo mental para cerrar la brecha que existe del dominio del problema, el qué, al dominio de la solución, el cómo. Este proceso es plasmado mediante la generación de **Diagramas de Secuencia** para aquellas interacciones significativas desde un punto de vista técnico las cuales podrán ser anexadas al **Documento de Arquitectura**.
- El nivel de ceremonia estará acorde a la criticidad del sistema que se desarrolla. En casos minimalistas y donde se desee tener un mínimo *overhead* metodológico el diseño podrá ser realizado en papel o en un pizarrón, en casos de mayor ceremonia se utilizarán herramientas de modelado visual que ayuden en este proceso.

Una función esencial es la de asignación de funcionalidad en que el Líder de Proyecto o el Arquitecto se encargarán de nombrar a un responsable por cada unidad de funcionalidad especificada. Puede tratarse de un caso de uso, de un grupo de clases, de un paquete. Lo importante es que exista un responsable para cada artefacto que mantenga actualizada **la trazabilidad requerida por la metodología propuesta** pudiendo realizar cualquier tipo de modificación sobre el mismo y manteniendo un adecuado control de cambios.

Una vez finalizado el proceso de diseño detallado de las clases a ser construidas durante la iteración, se procede a la siguiente disciplina que es la de Implementación- Pruebas.

4.1.5.4 IMPLEMENTACIÓN – PRUEBAS

En la disciplina de Implementación-Pruebas se lleva a cabo la producción del software. El equipo de desarrollo se encarga de implementar la funcionalidad especificada en los casos de uso mediante el lenguaje de programación elegido.

Dado el énfasis que la metodología propuesta propone en reducir la brecha comunicacional entre las personas que forman el equipo, esto mismo repercute en la preferencia de elegir para proyectos de esta envergadura aquellos lenguajes de programación que minimicen la brecha entre la realidad y la realización de la misma en un lenguaje. Idealmente, la metodología propuesta fomenta el uso de frameworks o tecnologías bajo el paradigma de orientación a objetos, como Power Builder, Java, C, MS. NET o Smalltalk, los cuales permiten el máximo de eficiencia por línea de código para aplicaciones de IS/IT, los lenguajes de este tipo permiten una menor cantidad de líneas de código por puntos de función, maximizando así la productividad del desarrollador.

En paralelo a la construcción de los componentes se realiza la revisión de los mismos mediante la prueba, el cual tiene una gran relevancia en la metodología propuesta. Siendo está una de las maneras en que se llevan a cabo los mecanismos de SQC [Software Quality Control], la metodología define diversas pruebas que podrán ser realizadas en las iteraciones. Las mismas incluyen:

4.1.5.4.1 PRUEBAS UNITARIAS

Definición: las Pruebas Unitarias son pruebas realizadas a nivel de las clases y métodos construidos para la aplicación. Estas pruebas son implementadas por los mismos programadores en el lenguaje de programación utilizando clases de prueba encargadas de verificar el comportamiento correcto de los métodos en una clase. Para ello es conveniente contar con algún framework de pruebas unitario que le facilite al equipo de desarrollo la creación, mantenimiento y ejecución de una suite de pruebas automatizadas.

Alcance: las mismas son utilizadas para controlar la calidad del código construido y son ideales para las pruebas de integración y de regresión.

4.1.5.4.2 PRUEBAS FUNCIONALES

Definición: las Pruebas Funcionales verifican el flujo de interacción de la funcionalidad definida en los casos de uso. Esta funcionalidad – que está escrita en forma narrativa en los casos de uso y es plasmada mediante la interacción de objetos enviándose mensajes entre sí en los diagramas de secuencia – debe ser especificada en pruebas automatizadas que serán creadas por el usuario en conjunción con la ayuda técnica del analista de pruebas y serán ejecutadas por el analista de pruebas durante cada iteración. Cabe remarcar que debe ser el usuario el que define el contenido de las mismas ya que de esta forma proveen el feedback que permite al equipo de desarrollo continuar las iteraciones asegurándose de la calidad del producto construido.

Alcance: las mismas son utilizadas durante todo el desarrollo y su alcance está definido por el conjunto completo de requerimientos funcionales definidos en los casos de uso.

4.1.5.4.3 PRUEBAS DE ACEPTACIÓN

Definición: el objetivo de las Pruebas de Aceptación es la verificación que el software construido en las iteraciones cumple con las funcionalidades solicitadas por el usuario y está listo para ser utilizado en el ambiente del cliente. Estas pruebas revisten gran relevancia porque implican que el cliente da conformidad respecto al sistema desarrollado y lo acepta.

Alcance: las mismas son realizadas durante las Iteraciones con entregables en las que se lleva a cabo la transición de la aplicación al ambiente de producción en el cliente.

4.1.5.5 DESPLIEGUE

La disciplina de Despliegue permite que el sistema construido sea transferido al ambiente de producción para ser utilizado por el usuario. Esto

no significa que el software debe estar implementado en su totalidad, cubriendo el 100% de los aspectos funcionales y no funcionales sino que en una iteración determinada se desee liberar una versión con un porcentaje de los casos de uso implementados. Dado el esquema de desarrollo iterativo e incremental, durante las iteraciones ya mencionadas se va agregando funcionalidad y se hace "crecer" al sistema hasta que cumpla con los requerimientos que se especificaron al principio, sumados a los cambios surgidos a lo largo del desarrollo.

Durante esta disciplina, se deben realizar actividades tendientes a hacer una entrega completa con una funcionalidad previamente determinada la cual será desplegada en producción. Las actividades incluidas dentro de esta disciplina son:

- Generar Manuales del Usuario, de Operación, etc.
- Realizar un empaquetamiento de componentes junto con scripts de instalación que el Cliente utilice para instalar la aplicación en su entorno.
- Ejecutar el conjunto completo de pruebas funcionales creadas durante el desarrollo, hasta obtener la aceptación del Cliente.
- En caso de ser necesario, definir las actividades de migración al nuevo sistema.
- Capacitar a los Usuarios que utilizarán el nuevo sistema.
- Generar la Nota de Entrega con la totalidad de los artefactos que se le entregan al Cliente [con su correspondiente versionado] al final de la disciplina.

El Líder de Proyecto deberá encargarse de lograr el consenso de todos los involucrados en una fecha de entrega del sistema. Para esto coordinará las fechas de forma que el Cliente tenga la oportunidad de verificar el correcto funcionamiento del sistema, mediante las pruebas de aceptación que se han generado hasta el momento. Es conveniente, que se planifique una reunión formal entre las dos partes [Cliente y Equipo de Desarrollo] para que todos estén en conocimiento de los resultados obtenidos, se puedan

realizar comentarios respecto a las cuestiones que se deberían mejorar en el proceso para las siguientes iteraciones. En otras palabras, esta reunión serviría como una revisión Post-Mortem en la que se evaluarían todas las iteraciones de desarrollo que precedieron a la actual, y cuáles fueron las actividades que funcionaron exitosamente y cuáles fueron las dificultades encontradas en el camino.

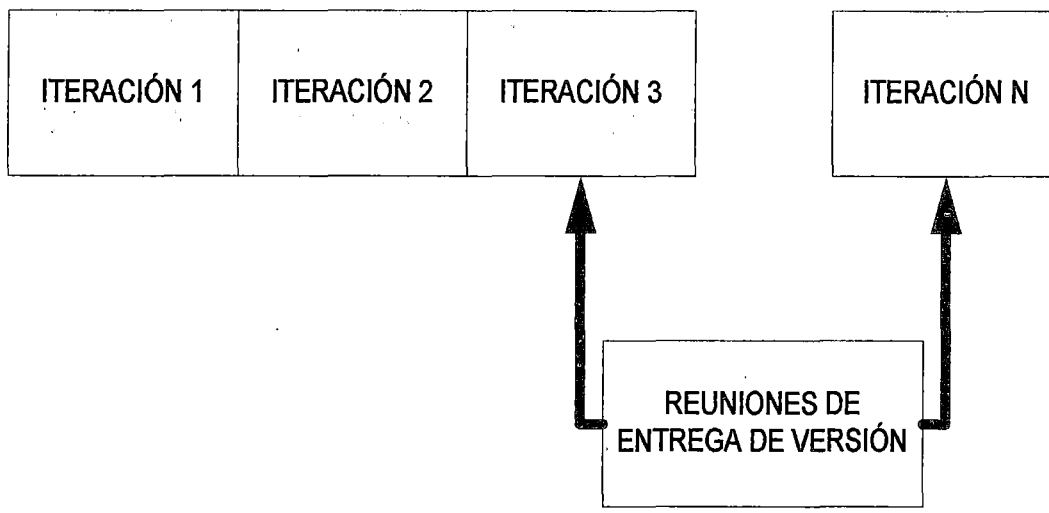


Figura 4.9: Esquema cronológico de reuniones de entrega de una versión del sistema.

Como se observa en la Figura 4.9, las Reuniones de Entrega de Versión se realizan al final de cada iteración de release. En este caso, la Iteración 2 va a ser la primera entrega de funcionalidad en el proyecto. Toda la funcionalidad desarrollada a lo largo de las dos primeras iteraciones será liberada al Cliente al fin de la Iteración 2. Por ello, se planifica la Reunión de Entrega de Versión al final de la misma. Este caso es equivalente a lo que ocurre en la Iteración N.

4.1.6 DISCIPLINA DE SOPORTE

Las disciplinas de soporte ocurren a lo largo de todo el ciclo de vida del proyecto y dan soporte a las actividades relacionadas con la construcción del software. Las mismas son tan importantes como las disciplinas ya

mencionadas pero la diferencia es que sirven propósitos organizacionales no directamente relacionados con la producción de sistemas. Entre estas nuevas disciplinas se mencionan cuatro, ver Figura 4.10.

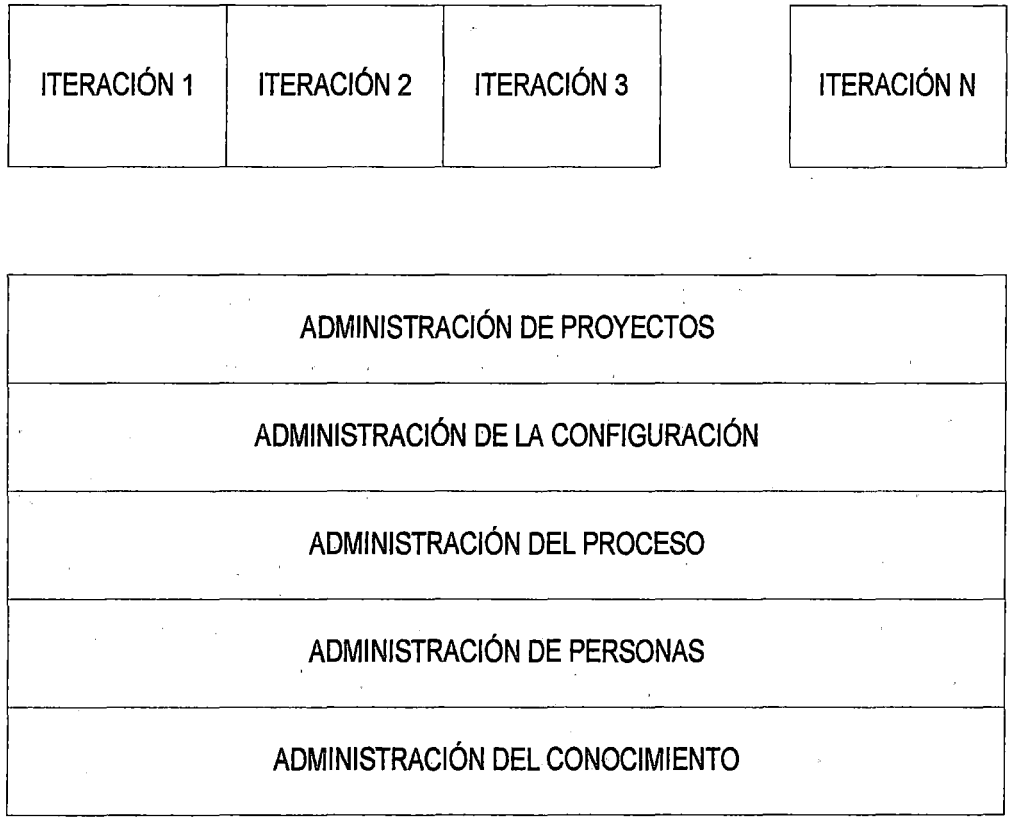


Figura 4.10: Disciplinas de Soporte de Metodología propuesta.

4.1.6.1 ADMINISTRACIÓN DE PROYECTO

La disciplina de Administración de Proyecto engloba todas las actividades relacionadas con la gestión del emprendimiento.

- Dentro de la metodología propuesta esto refiere a las tareas que el Líder de Proyecto lleva a cabo para garantizar un ambiente saludable de trabajo:
  - Una medición del grado de avance del proyecto.
  - Un continuo monitoreo y mitigación de riesgos.
- Estas actividades permiten que el proyecto llegue a su éxito, entregando la aplicación:

- En tiempo.
- Forma.
- Dentro de los costos presupuestados.

- Dentro de la metodología propuesta el rol del Líder de Proyecto es bastante distinto a las metodologías tradicionales basadas en la idea del control jerárquico. Bajo este paradigma el Líder era quien repartía el trabajo a las personas y quien estimaba el tiempo que llevarían todas las actividades mediante la realización de un WBS[24] detallado y continuamente monitoreaba el trabajo en relación a sus estimaciones para aplicar correcciones.
- Bajo la metodología propuesta el Líder no es más que la persona que:
  - Habilita al equipo de desarrollo para que pueda trabajar cómodamente, sin obstáculos, resolviendo las necesidades que ésta tenga.
  - Tiene el carácter y las responsabilidades de un facilitador.
  - Asimismo, es la cara visible del equipo hacia los del proyecto – salvando los stakeholders claves que se mencionan posteriormente.
  - Deberá reportar y conocer el status del grupo y cómo el desarrollo se va dando en relación a la planificación establecida.
- La metodología propuesta recomienda que se establezca una **planificación inicial del proyecto** que permita tener objetivos claros respecto a los grandes hitos para obtener feedback del cliente respecto al avance realizado.
- Se armará un **plan de proyecto** con las fases y las iteraciones durante las que se irá construyendo el sistema.

---

[24] Work Breakdown Structure en español es Estructura de Descomposición de Trabajo.

- Dadas las características de este proceso ágil, no se debería realizar un plan ultra detallado con un WBS que presente todas las tareas a realizarse. Simplemente, se fraccionará la funcionalidad relevada hasta el momento en iteraciones en las cuales se irán liberando versiones al cliente para obtener el tan valioso feedback.
- Es importante destacar que el plan es armado consultando al equipo de desarrollo; la única tarea del Líder de Proyecto es armarlo en algún formato presentable al cliente, como por ejemplo un diagrama Gantt o una planilla Excel, dependiendo del nivel de formalidad requerido.
- El equipo de desarrollo será el que estimará la duración de las tareas a este nivel, entendiendo que la estimación será de una precisión muy limitada, ya que no se cuenta con mucha información de la funcionalidad a desarrollar. En cada iteración el plan será actualizado para reflejar la realidad.
- Una de las prácticas importantes es la de llevar a cabo una reunión diaria de no más de 15 minutos emulando a la Scrum Daily Meeting recomendada por Scrum. La misma refuerza el patrón de Máxima Comunicación que describe la necesidad de que el flujo de interacción sea frecuente; en este caso para que el Líder de Proyecto pueda medir el avance realizado diariamente, resolviendo cualquier problemática del equipo que este a su alcance y eliminando obstáculos. El cliente podrá participar de la misma pero sin poder esgrimir comentarios, sólo como un oyente pasivo.

4.1.6.2 ADMINISTRACIÓN DE LA CONFIGURACIÓN

La disciplina de Administración de la Configuración contiene las actividades relacionadas:

- Con la administración del repositorio que almacena los ítems de configuración generados en un proyecto. Esta es un área de

conocimiento dentro de la Ingeniería de Software en la que mayores avances se han realizado y que ha estado presente en los proyectos de software desde aproximadamente 1980. Actualmente la mayor parte de los desarrollos incluyen alguna herramienta de manejo de la herramienta Gestión de Configuración del Software que automatiza todas las operaciones sobre el versionado.

- La metodología propuesta recomienda colocar todos los artefactos generados o consumidos en un proyecto bajo control de configuración. Esto permitirá que en cualquier momento se pueda recuperar una versión determinada de un ítem o ítems de configuración para recrear la cronología del proyecto o volver cambios atrás, o llevar a cabos registros de trazabilidad y/o auditorias entre otras cosas.
- En particular, el uso de una herramienta de Gestión de Configuración del Software le da la posibilidad al equipo de desarrollo de tener prácticas eficientes al momento de realizar las actividades en un proyecto. Entre estas prácticas cotidianas presentadas en cualquier desarrollo se pueden mencionar:
    - Los desarrolladores pueden trabajar en un mismo proyecto, compartiendo todos los ítems de configuración.
    - Los desarrolladores pueden compartir el esfuerzo en el desarrollo de cualquier artefacto, sea un objeto, un documento, etc.
    - Los desarrolladores pueden acceder la versión estable actual del sistema para verificar si el código que generaron seguirá funcionando cuando sea integrado.
    - Los desarrolladores pueden volver a una versión anterior del sistema para hacer pruebas contra esta.
    - Los desarrolladores pueden trabajar en distintas ramas en paralelo sobre los artefactos del sistema, permitiendo

tener una rama estable, otra para experimentar, y así sucesivamente.

- Es importante que las herramientas sean usadas consistentemente y que todo el equipo tenga el conocimiento para trabajar con ellas. Esto podrá requerir capacitación a las personas que lo utilizarán.
- Dentro del marco ágil de la metodología propuesta la práctica utilizada dentro de esta disciplina deriva de una práctica propuesta por XP denominada ***Collective Code Ownership***. Está práctica recomienda que no existan "dueños" de los ítems de configuración subidos al repositorio sino que cada persona podrá interactuar con cualquier ítem, pudiendo modificarlo y subir una nueva versión. En cierto sentido el equipo de desarrollo es responsable por todo el sistema por lo tanto todos los miembros de este son "dueños" de todos los ítems. Esto es posible gracias a la fluida comunicación existente entre las personas que fomenta la metodología propuesta y al grado de responsabilidad que tienen las personas.
- Dentro de esta disciplina se incluyen aquellas actividades relacionadas con la ***Administración de Cambio***. Es decir, un proceso formal que nos permita controlar la forma en que los cambios son implementados durante el desarrollo. Dadas las características de la metodología propuesta, los cambios son tomados como una posibilidad de tener un mejor entendimiento de la aplicación a construir por lo cual serán priorizados por el Cliente de acuerdo a su valor del negocio e implementados según corresponda en una iteración determinada. Dependiendo del costo asociado a los mismos el Cliente puede decidir descartarlos. Es de suma utilidad disponer de una herramienta que automatice el seguimiento de los cambios a lo largo de su

ciclo de vida, notificando al equipo de desarrollo de los cambios de estado de cada uno a lo largo del proyecto.

4.1.6.3 ADMINISTRACIÓN DEL PROCESO

La disciplina de Administración del Proceso permite que la metodología propuesta sea adaptada a las necesidades de las personas y del proyecto en cuestión. Se decidió incluirla como una disciplina separada. El Proceso Unificado de Rational [RUP, 2002] incluye a las actividades de personalización de proceso dentro de la disciplina de Ambiente en la que también aparecen todas las consideraciones humanas. Cabe mencionar, que la metodología propuesta se divide en dos partes.

- Primero, Administración de Proceso.
- Segundo, Administración de Personas.

Una vez que la fábrica de software ya posee una metodología como la metodología propuesta estandarizada en sus proyectos y conocida por todas las personas, la misma deberá ser adaptada de acuerdo al contexto de cada proyecto en que se utiliza.

- Es decir, habrá distintos proyectos con características dispares en cuanto a:
  - Tamaño del equipo.
  - Grado de criticidad del sistema.
  - Stakeholders involucrados.
- Todos estos factores darán como resultado una necesidad metodológica única, con ciertos roles, ciertos artefactos, ciertas actividades, etc.
- Consecuentemente la metodología propuesta dispone de una disciplina que permite adaptarla a las necesidades de la instancia de proceso en que se esté.
- Será clave en esta disciplina la figura del Coordinador quien empezará con un análisis del proyecto, tomando la planificación original, la funcionalidad, el equipo de trabajo, el perfil del

cliente, y con todo esto definirá como se empleará la metodología propuesta en dicho proyecto.

- El Coordinador deberá tener un amplio conocimiento del espectro metodológico existente para poder extraer aquellos elementos de la metodología propuesta que considere que no apliquen y para poder agregar elementos que no estén contemplados y que formen parte de otros procesos.
    - En este sentido se puede recomendar una profunda capacitación en el RUP que muchas veces cumple la función de un metaproceso sirviendo como base de conocimiento para los ingenieros de software.
- Cabe remarcar que la tarea de adaptación del proceso es llevada a cabo por el Coordinador en cooperación con el equipo de desarrollo. Esto resulta muy relevante ya que será este ultimo el usuario final de la metodología por lo cual el Coordinador consultará con los miembros más especializados para entender los mecanismos de desarrollo involucrados.
- Una vez que la metodología haya sido configurada para un proyecto, se harán reuniones esporádicas [una buena práctica consiste en coordinarlas con el fin de cada iteración] en las que se evaluará la adecuación de la metodología propuesta a las características del proyecto. Se analizarán los resultados positivos y negativos, pudiendo el Coordinador hacer cambios sobre la marcha para mejorar la forma de trabajo.
- Finalmente, cuando el proyecto esté llegando a su fin, el Coordinador, con la ayuda del Administrador de Conocimiento, almacenará la información acerca de cómo el proceso fue usado, con el propósito de mantener un historial de adaptaciones realizadas por tipo de proyecto.

4.1.6.4 ADMINISTRACIÓN DE PERSONAS

La última de las disciplinas recomendadas por la metodología propuesta es la de Administración de Personas que agrupa todas aquellas actividades relacionadas con los aspectos humanos del desarrollo.

- Nuevamente la figura clave de esta disciplina será el Coordinador, con una importante cooperación del Líder de Proyecto.
- Una de las primeras actividades dentro de esta disciplina es la de nivelar a los miembros del equipo de desarrollo. Es importante que cada uno tenga todos los conocimientos necesarios para encarar las actividades por las que es responsable.
- El Coordinador deberá llevar a cabo talleres de capacitación y mentoring [25] hasta asegurarse de que las personas han aprendido.
- Esta capacitación refiere:
    - Aspectos técnicos.
    - Demás disciplinas de Metodología propuesta, incluyendo la descripción del proceso.
- Otra de las actividades que será realizada en forma conjunta con el Líder de Proyecto es mantener la salud del equipo de desarrollo. En otras palabras:
    - Velar porque la motivación del equipo sea alta.
    - Garantizar relaciones interpersonales positivas.
    - Minimizar las fricciones debidas a distintos tipos de personalidad.

---

[25] El mentoring es un proceso de aprendizaje personal por el que una persona asume la propiedad y la responsabilidad de su propio desarrollo personal y profesional.

Para ello, se establece una relación personalizada dirigida por el mentorizado a través de la cual el mentor invierte su tiempo, comparte su conocimiento y dedica su esfuerzo para que el mentorizado disponga de nuevas perspectivas, enriquezca su forma de pensar y desarrolle todo su potencial como persona y como profesional.

- Entender a cada individuo para poder sacar lo mejor de sí mismo.

- Existe un solapamiento entre esta disciplina y la de Administración de Proyectos en este punto. Se destaca que el mismo no es tan importante desde un punto de vista conceptual pero la diferencia está marcada en que las actividades de esta disciplina tienen como objetivo el bienestar de las personas en el proyecto mientras que el management[26] se relaciona con la consecución exitosa del proyecto.
- La práctica recomendada para esta disciplina tiene que ver con el patrón de Máxima Comunicación. Es decir, si se tienen canales de comunicación suficientemente ricos las dificultades podrán ser resueltas, en la mayoría de los casos, utilizando el sentido común y la buena predisposición de la gente.

4.1.6.5 ADMINISTRACIÓN DEL CONOCIMIENTO

Un proceso de desarrollo de software debe proveer un mecanismo que permita que el conocimiento generado en un proyecto sea mantenido dentro de la organización. En la industria de software moderna es vital poseer mecanismos para reutilizar el conocimiento que se va generando a medida que la organización va ejecutando distintos tipos de proyectos.

- Para ello se definen los tres niveles de refinamiento hasta llegar al conocimiento; son: datos, información y conocimiento. Comenzando con el nivel inferior:
  - Los datos consisten en valores discretos y objetivos acerca de eventos, pero sin ningún contenido acerca de su importancia o relevancia; a partir de éstos se puede generar información.

[26] Peter F Drucker – "Management es un órgano multipropósito que maneja un negocio y maneja gerentes y manejar trabajadores y trabajo".

- La información son datos organizados de forma de servir de utilidad para las personas que mediante el contexto la utilizan para encarar tareas o tomar decisiones.
- Finalmente, el conocimiento requiere entender la información y tiene que ver con la relación entre ítems de información, su clasificación y su meta-dato [información de la información]. La experiencia es conocimiento aplicado.

- Se propone la utilización de una fábrica de conocimiento ágil siguiendo la línea de teoría de Victor Basili [Basili, 1990]. La misma consiste en una estructura paralela al equipo de desarrollo, la cual tiene personas de dedicación exclusiva dedicadas a capturar, empaquetar, almacenar y distribuir a lo largo de la organización.
- A partir de esta idea y su adecuación dentro de una metodología ágil en que se tiene pequeños grupos de trabajo, la metodología propuesta define un rol específico que es el Administrador del Conocimiento. Éste podrá ser desempeñado en forma rotativa por diferentes personas dentro del equipo de trabajo.
  - Su tarea consiste en documentar todo aquel conocimiento novedoso que haya sigo generado durante el proyecto para que pueda ser reutilizado por otras personas.
- En un contexto ágil, es imperativo que este patrón no genere una extrema burocracia dentro de la organización del proyecto. Por eso se definió un rol particular como ya fue mencionado el cual deberá entender la arquitectura del sistema y los aspectos funcionales más importantes para poder capturarlos en documentos que puedan ser recuperados y consumidos en el futuro. Esto es de suma importancia y tiene que ver con el aprendizaje organizacional sugerido por Tom DeMarco [DeMarco, 1999]. Asimismo se recomienda capturar todas

aquellas experiencias humanas positivas y negativas que se tuvieron en el proyecto para poder agruparlas en formato de patrones y anti-patrones que puedan ser dispersados y aprendidos por todas las personas.

- Un comentario muy importante en relación a la Administración del Conocimiento es la necesidad de invertir en la gente y en abocar por la motivación de los equipos de desarrollo.
    - Si una organización no valora a las personas que la conforman, nunca podrá aprender.
    - Si en una organización la gente trabaja poco tiempo y se va, habiendo mucho nivel de rotación, el aprendizaje será escaso.
    - Puesto de otra forma por Tom DeMarco: El aprendizaje está limitado por la habilidad de una organización de mantener a su gente.

4.1.7 ARTEFACTOS

Las metodologías ágiles en su mayoría tienden a minimizar la cantidad de artefactos generados en un proyecto. El énfasis está puesto en el código entregado y cualquier documento construido es visto como un costo indirecto. Sin embargo, existen muchos artefactos necesarios para propósitos de comunicación, diseño, gestión sin los cuales un proyecto no llegaría al éxito. La metodología propuesta enumera algunos artefactos que considera requisitos mínimos para tener madurez en el proceso de desarrollo.

4.1.7.1 DOCUMENTO DE VISIÓN

Definición: el Documento de Visión es realizado durante la fase de Concepción y sirve como contrato entre el Cliente y el Equipo de Desarrollo respecto a lo que se va a construir. En la Visión deberán identificarse:

- Los stakeholders del proyecto.
- La totalidad de las características del sistema a ser construido.

- Los requerimientos suplementarios que se detectaron en forma temprana.
- En forma optativa podrá incluirse un resumen de los casos de uso críticos del aplicativo.

Responsable: la Visión es construida por un Analista Funcional en conjunto con el Líder de Proyecto. La misma debe ser mantenida para reflejar los cambios al alcance durante el proyecto.

4.1.7.2 PLAN DE PROYECTO

Definición: el Plan de Proyecto es un documento mantenido por el Líder de Proyecto con toda la información de gestión.

- El mismo suele incluir un Gantt con el cronograma y las tareas del proyecto.
- Dependiendo del nivel de formalidad se generará un WBS y/o algún otro tipo de plan a incluirse dentro de este [plan de pruebas, de comunicaciones, de administración de requerimientos, de administración de cambios, etc.].
- El Plan de Proyecto es creado en forma temprana durante la Concepción y actualizado durante las fases posteriores.

Responsable: el Líder de Proyecto es el responsable de este documento y deberá mantenerlo actualizado hasta la terminación del proyecto.

4.1.7.3 LISTA DE RIESGOS

Definición: la Lista de Riesgos se utiliza para capturar los riesgos más relevantes que se presentan en el proyecto, y para poder monitorearlos, asignarles prioridad, impacto y probabilidad de ocurrencia. Sirve para planificar las iteraciones y para tener en vista aquellos riesgos que, por su criticidad, deberán ser mitigados lo más tempranamente posible.

Responsable: el Líder de Proyecto es el responsable y el principal consumidor de este documento y deberá mantenerlo actualizado hasta la terminación del proyecto.

4.1.7.4 MODELO DE CASOS DE USO Y ESPECIFICACIONES DE CASOS DE USO

Definición: los Casos de Uso se utilizarán para especificar los requerimientos funcionales de la aplicación. Los mismos servirán para guiar los demás artefactos y para la construcción de los componentes de la aplicación. Asimismo, la metodología propuesta recomienda la generación del Modelo de Casos de Uso para tener una visualización gráfica del universo funcional de la aplicación y poder comunicarla tanto internamente como al Cliente para su validación. Los Casos de Uso y el Modelo serán creados en las primeras fases del proyecto, aunque podrán también ser especificados en las iteraciones de construcción.

Responsable: los Analistas son responsables de la creación y el mantenimiento de estos artefactos, los cuales serán consumidos por el resto del equipo de desarrollo.

4.1.7.5 DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE [SRS]

Definición: el documento de especificación de requerimientos de software se utiliza para:

- Especificar los requerimientos funcionales de carácter más técnico de la aplicación.
- También en este se incluirían los requerimientos no funcionales, las reglas de negocio, y las restricciones que se aplicarán a la solución.
- En este documento se pondrán todos los requerimientos que queden fuera de los casos de uso.

- La idea es tener algún tipo de constancia documental de todos aquellos requerimientos de carácter técnico que debe contemplar la solución.

Responsable: los Analistas son responsables de la creación y el mantenimiento de este artefacto, el cual serán consumidos por el resto del equipo de desarrollo.

4.1.7.6 DESCRIPCIÓN DE LA ARQUITECTURA

Definición: el documento de Descripción de la Arquitectura especifica los aspectos técnicos de la solución propuesta por el equipo de desarrollo. Sirve como medio de comunicación entre el Arquitecto y el equipo en relación a cómo deberán ser implementados los casos de uso de la aplicación.

Responsable: el Arquitecto es el principal responsable por la realización de este artefacto, así como la creación de un prototipo ejecutable [Prueba de Concepto] que valide que la misma cumpla los requerimientos no funcionales y las cualidades sistémicas que hacen a los atributos de calidad no relacionados directamente con las necesidades del usuario.

4.1.7.7 CASOS DE PRUEBA

Definición: los Casos de Prueba contienen la especificación de cómo será validado el sistema. Estos son realizados basándose en los casos de uso y planteando todos los escenarios posibles que éstos pueden contener. Dado que puede ser bastante burocrático realizar la totalidad de Casos de Prueba existentes en un sistema, se recomienda generar los más importantes y después anotar las variantes en un Caso de Prueba Estándar.

Responsable: el analista de pruebas es el principal responsable por la realización de este artefacto. El analista de prueba será el encargado del diseño y la ejecución de los Casos de Prueba.

4.1.7.8 SCRIPTS DE DESPLIEGUE

Definición: los Scripts de Despliegue son necesarios para la instalación del producto en un entorno determinado. El mismo automatiza las tareas de empaquetamiento, versionado, compilación, etc.

Responsable: algún Programador será responsable de la generación y mantenimiento de estos scripts.

4.1.7.9 PLANILLA DE INCIDENTES

Definición: la Planilla de Incidentes será utilizada para registrar y tener seguimiento de aquellos errores, mejoras, tareas que el analista de pruebas o alguna persona de aseguramiento de la calidad necesitan reportar. Son los denominados sistemas de seguimiento de incidentes que pueden ser implementados manualmente o mediante alguna herramienta.

Responsable: la Planilla de Incidentes será utilizada por todos los miembros del equipo. En particular será muy utilizada por los analistas de pruebas quienes cargarán todos los errores encontrados durante la ejecución de los Casos de Prueba los cuales serán leídos y corregidos por los Programadores.

4.1.7.10 REPOSITORIO DEL PROYECTO

Definición: el Repositorio es la herramienta fundamental para cubrir la disciplina de Administración de la Configuración. En el repositorio se almacenan todas las versiones de todos los archivos y directorios del proyecto. La metodología propuesta recomienda que todo artefacto generado durante un proyecto sea puesto bajo control de versiones y no sólo el código fuente de la aplicación. Esto permitirá en cualquier momento poder comparar versiones, volver a recuperar versiones anteriores, y generar ramas de desarrollo paralelo.

Responsable: algún Programador o alguna persona de Infraestructura de la organización será responsable de la creación, mantenimiento y eliminación [en casos que se desee dar de baja el proyecto] del repositorio.

4.1.7.11 NOTA DE ENTREGA

Definición: la Nota de Entrega sirve para especificar aquello que se le entrega al Cliente en un entregable determinado de la aplicación. En la misma se constatará el número de versión, los cambios de la versión anterior, los errores conocidos y las mejoras efectuadas.

Responsable: cualquier miembro del equipo de desarrollo que vaya a entregar algún artefacto al Cliente deberá crear la correspondiente Nota de Entrega.

4.2 PATRONES DE DESARROLLO RECOMENDADOS

Los patrones surgieron del universo de la arquitectura del trabajo de Christopher Alexander durante los años 1977-1979. En su primer libro, [Alexander, 1977] menciona: "Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y después describe la base de la solución a dicho problema de una forma que se puede utilizar dicha solución un millón de veces sin realizarla dos veces de la misma manera". Posteriormente [Alexander, 1977] simplificó la definición teniendo en cuenta que un patrón plantea un problema y después propone una solución al mismo. También nos da un contexto para entender cuándo la solución debe ser aplicada. Los patrones de [Alexander, 1977] no tuvieron una influencia muy importante en la comunidad de arquitectura, pero sí tuvieron un gran impacto en la industria del software alrededor de 1995, cuando se editó el libro de Gang Of Four [Gamma, 1995]. Basado en estos conceptos y en [Shenone, 2004] se define los patrones que la metodología propuesta recomienda en el desarrollo de software.

Dichos patrones son los mencionados a continuación:

- Máxima Comunicación
- Comunicación Interna al Equipo
- Comunicación Externa al Equipo
- Participación Activa del Cliente
- Estimaciones Ágiles
- Enfoque en la Arquitectura

- Integración Continua
- Enfocado a la Persona
- Calidad en el Proceso

4.2.1 MÁXIMA COMUNICACIÓN

Las metodologías ágiles solo tienen sentido si existe comunicación entre las personas. Esto no solo refiere a las personas del equipo de desarrollo, sino a todas las personas que se vean afectadas en distinta medida por la ejecución del proyecto; los denominados durante esta tesis como stakeholders.

Bajo el patrón de Máxima Comunicación se encuentra uno de los principios esenciales de la metodología propuesta. Relacionado estrechamente con una de las actividades más comunes de los seres humanos: la comunicación – que representa las diversas formas de transmisión de datos/información/conocimiento/experiencia entre individuos.

El software se encuentra entre los elementos de mayor complejidad creados por el hombre. Dada esta complejidad inherente a su esencia, el software presenta una gran dificultad en su construcción caracterizada por la aleatoriedad de las posibilidades en el desarrollo de cualquier sistema y la incapacidad de concebir procesos repetibles – como ocurre en las demás ingenierías. Si bien se han logrado muchos avances en cuanto al manejo de dicha complejidad, la misma aún persiste y genera resultados no deseados como cronogramas excedidos, baja calidad o funcionalidad incompleta.

Existen dos tipos de comunicaciones que este proceso toma en cuenta para mejorar el desarrollo. Estas comunicaciones difieren en relación a las personas involucradas en las mismas.

- La primera es la comunicación interna, es decir la comunicación entre todos los integrantes del equipo de desarrollo – desde el líder del proyecto hasta el programador.
- La segunda comunicación a la que nos referiremos es la comunicación entre los integrantes del equipo de desarrollo

y el cliente o usuario del sistema. Es decir, todas las comunicaciones que el equipo mantenga con los stakeholders. En la figura 4.11, se tiene una representación gráfica de los niveles presentados en este párrafo.

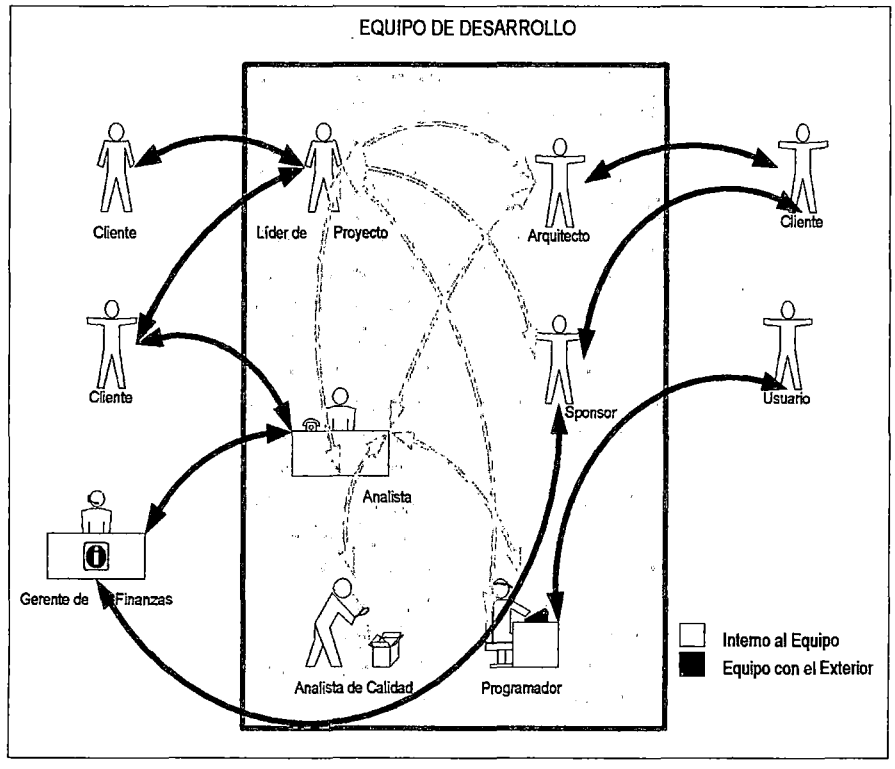


Figura 4.11: Comunicaciones existentes en un proyecto.

Esta división permite concentrarnos en las prácticas que podrán ser aplicadas en cada caso. Analizaremos las formas de atacar el desarrollo de manera de maximizar la comunicación entre las partes involucradas acorde al contexto en que se plantean.

Tomando los principios que guían a la metodología propuesta, continuamente se priorizan las personas sobre el proceso. La metodología debe proveer mecanismos que fomenten las interacciones ayudando a mejorar la motivación y la productividad del equipo de desarrollo. De esta forma el flujo de información entre las mentes de los desarrolladores será maximizado permitiendo mayor progreso en el proyecto. Alistair Cockburn

[Cockburn, 2001a] analizó este fenómeno en detalle evaluando el impacto de distintos modelos comunicacionales. La Figura 4.12 muestra la influencia en la efectividad de la comunicación de acuerdo al canal que se utiliza para establecer la misma.

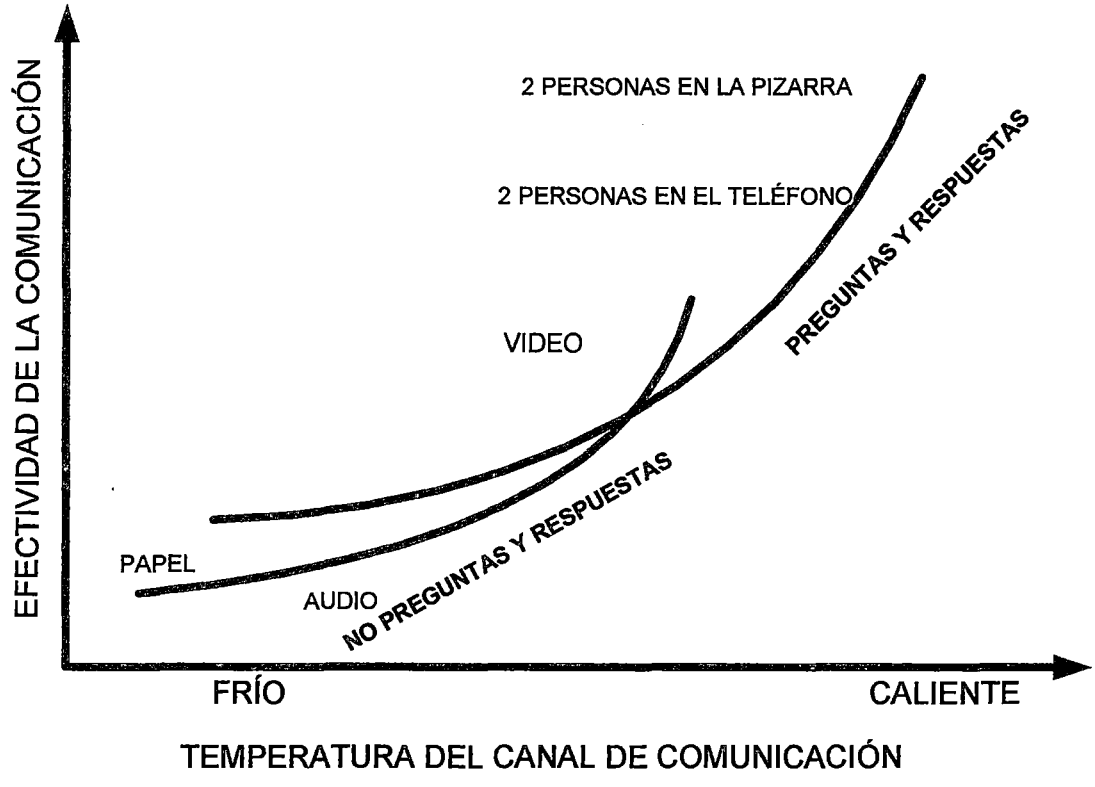


Figura 4.12: Efectividad de distintos modos de comunicación. Tomada de [Cockburn, 2001a].

Acorde a los resultados reportados en dicha investigación, la metodología propuesta recomienda diversas prácticas que estimulen canales de comunicación más abiertos tanto internamente al equipo como externamente con los stakeholders del proyecto.

4.2.2 COMUNICACIÓN INTERNA

Existen los patrones siguientes:

- La primera recomendación tiene que ver con el patrón de **Mínima Dispersión Física**. Según este patrón:

- El equipo de desarrollo debe estar colocado físicamente en una única oficina que tenga una distribución centralizada al momento de trabajo con lugares privados para cuestiones extra laborales.
- Esta es la propuesta de XP la cual no hace más que obedecer al canal de comunicación que es ver a las personas cara a cara para interactuar con ellas.

- Asimismo otro patrón es el de **Interrupciones Mínimas**, el cual se refiere a la necesidad de liberar al equipo de desarrollo de interrupciones referidas a otros ámbitos. En principio, se recomienda que la oficina del equipo de desarrollo este separada mediante paredes del resto de la organización. Si esto no se logra, se tendrán flujos de información que no refieren al desarrollo y que empeorarán la calidad del ambiente diseñado para maximizar la comunicación – como planteaba el primer patrón mencionado. Tom DeMarco [DeMarco, 1987] indaga bastante sobre los efectos nocivos de mantener entornos de trabajo con mucho ruido e información innecesaria que atenta contra el trabajo de las personas. También este patrón aplica a la necesidad de que las personas no estén continuamente interrumpiéndose entre sí, esto es evidente en los casos de una persona que juega un rol clave y que constantemente debe resolver dudas. La idea subyacente es la de tratar primero de evacuar las dudas mediante lectura de algún documento, si esto no funciona, se podrá pedir auxilio a los pares. Como último recurso se debería convocar a esta persona, en algunos casos se pueden juntar un número importante de dudas e ir anotándolas y preguntarlas todas de una única vez. Sin embargo, dado que la comunicación es fomentada la metodología propuesta recomienda una interrupción antes que guardarse la pregunta y hacer alguna suposición al respecto.

- Un tercer patrón que se recomienda es el que Cockburn describe como **Radiadores de Información**. Según este patrón se recomienda tener las paredes de la oficina totalmente libres dispuestas para que se cuelguen en ellas pizarrones, hojas grandes, diagramas, notas post-it, etc. Estos actúan como radiadores de información ya que permiten que cualquier miembro del equipo pueda tener acceso a esa información constantemente sin necesidad de andar indagando a las demás personas. Más aún, los radiadores sirven para mantener información crítica a la vista de todos la cual puede ser discutida, analizada, modificada y comunicada continuamente. Ejemplos de información que se puede plasmar en estos es:
    - Diagramas de Arquitectura
    - Avance del proyecto en función de los casos de uso implementados
    - Resultado de la ejecución de los casos de prueba
    - Diagrama de casos de uso

Una implementación alternativa es tener un sitio web que disponga continuamente de información del proyecto. Adicionalmente, puede servir como base de conocimientos del proyecto y de la organización.

4.2.3 COMUNICACIÓN EXTERNA

Tan importante como la comunicación interna resulta la comunicación con los stakeholders del proyecto. Primero se debe dividir a los stakeholders en dos franjas: stakeholders claves para el éxito del proyecto y stakeholders normales.

- Los stakeholders claves serán aquellos:
    - Que poseen conocimiento del dominio del sistema a construir.
    - Son aquellos que aceptarán el sistema en cuestión si el mismo se adecúa a sus necesidades.
- Por otro lado, se tienen los stakeholders normales con los que el equipo interactúa ocasionalmente para algún propósito del

proyecto. Estos últimos juegan un rol que puede tener importancia para la salud del proyecto pero que no impacta directamente sobre la funcionalidad de la aplicación construida.

La metodología propuesta recomienda maximizar la comunicación con los stakeholders claves del proyecto. En el caso ideal se recomienda utilizar el patrón *On-Site Customer* derivado de XP que establece que durante todo el proyecto habrá un cliente real sentado en la misma oficina con el equipo de desarrollo, dispuesto a responder preguntas, resolver disputas, y priorizar las tareas llevadas a cabo. Cabe destacar que en este caso un "cliente real" refiere a una persona que utilizará el sistema una vez puesto en producción y que conozca la aplicación en su extensión para resolver distintas cuestiones que puedan darse.

Ocurre que en la mayoría de los casos los clientes tienen que hacer sus propios trabajos y no disponen de tiempo para estar continuamente formando parte del equipo de desarrollo. Dadas las realidades impuestas en la mayor parte de los proyectos, la metodología propuesta propone disponer de un cliente designado con el rol de Experto en el Dominio mencionado anteriormente. El mismo debe estar dispuesto a contestar cualquier pregunta del equipo de desarrollo ya sea mediante algún canal de comunicación más frío de acuerdo a la Figura 4.12 [ej.: teléfono, videoconferencia, mail, etc.] en un tiempo razonable que no debería superar algunas horas. En caso de que el problema requiera de una comunicación más personal, el cliente deberá disponer de tiempo para visitar al equipo de desarrollo y discutir el tema personalmente o bien para recibir a algún miembro clave en sus propias oficinas.

El énfasis está puesto en mantener el proceso lo más ágil posible, y dado que la recomendación es construir casos de uso que manifiesten las interacciones a un nivel no muy detallado siempre existirán dudas que podrán ser mitigadas por los Expertos en el Dominio.

Respecto a los stakeholders normales, la recomendación es que los mismos sean identificados en forma temprana y sus necesidades sean relevadas para verificar que el sistema conforme a estos adecuadamente. Si

bien no será necesario tener un canal abierto tan flexible la comunicación con estos debe ser fluida y honesta ya que en algún punto el sistema tendrá influencia sobre ellos o viceversa.

4.2.4 PARTICIPACIÓN ACTIVA DEL CLIENTE

La metodología propuesta plantea la necesidad de contar con una incondicional participación del Cliente durante el proyecto de desarrollo. Dadas las características de las aplicaciones a las que el proceso apunta, los únicos capaces de definir qué debe contener el producto son los usuarios dado que serán sus necesidades las que se verán afectadas. En esto la metodología propuesta es inflexible; no hay documento ni modelo ni herramienta que pueda sustituir el feedback provisto en forma directa mediante contacto cara a cara entre el analista y el cliente o usuario del sistema. No hay forma de alcanzar el éxito en un proyecto si no se cuenta con el compromiso y el soporte del sponsor. El apoyo de la alta dirección garantiza que los obstáculos que se presenten – políticos, económicos, sociales – puedan ser removidos mediante su activa participación.

Las razones de la inflexibilidad en este punto son aquellas que han revelado cuales eran los factores que más contribuían al fracaso de los proyectos de software. Según estudios realizados por el Grupo Standish [Standish, 1994] la falta de involucramiento por parte de los usuarios y el poco apoyo de la alta gerencia están entre los factores antes mencionados. Si se analiza este hecho se concluye que es lógico establecer que si el usuario no participa en la construcción del software nunca este último va a satisfacer sus necesidades por ser el usuario el único que las conoce.

En otras palabras, el no involucrar al usuario durante el desarrollo de software nos permite construir un producto de software que sólo cumple las necesidades de aquellas personas que no van a utilizarlo.

Para lograr esta activa participación del usuario se debe poder contar con usuarios expertos en el dominio que puedan prestar todo el tiempo necesario para las necesidades del equipo de proyecto. Entre estas se incluye la posibilidad de que los analistas puedan delinear requerimientos

con el grado de detalle necesario para que los programadores puedan comenzar con el diseño. También los programadores deben poder consultar a dichos usuarios si se les plantean dudas respecto al dominio del problema o existen omisiones o inconsistencias en los requerimientos.

Los usuarios deberán tener amplio conocimiento del dominio para poder resolver cualquier duda del equipo respecto a las funcionalidades a desarrollar. También deberá priorizar continuamente el valor del negocio de lo que se construye en cada iteración para guiar funcionalmente al equipo.

4.2.5 ESTIMACIONES ÁGILES

Al estar Orientadas a los Productos, las metodologías ágiles permiten realizar estimaciones de menor granularidad en el día a día para obtener más precisión en la estimación de la entrega de artefactos en cada iteración, en detrimento de una visibilidad a largo plazo. Sin embargo, es importante realizar estimaciones realistas de la duración de los proyectos sobre todo al principio de los mismos.

La técnica que propone la metodología propuesta es la estimación por puntos de caso de uso [*use case points*]. La misma mide a los sistemas desde una perspectiva funcional y es independiente de la tecnología. Sin tener consideración del lenguaje, método de desarrollo, o plataforma de hardware utilizada, el número de puntos de casos de uso de un sistema permanecerá constante. La única variable es la cantidad de esfuerzo necesario para desarrollar un conjunto dado de puntos de caso de uso; por lo tanto el Análisis por Puntos de Caso de uso puede ser utilizado para determinar si una herramienta, un ambiente, un lenguaje es más productivo comparado con otros dentro de una organización o entre organizaciones. Este es uno de los puntos críticos y uno de los valores más importantes de dicho análisis.

Dentro de los proyecto de implementación bajo el alcance de la metodología propuesta, el objetivo es poder hacer una estimación de la funcionalidad a ser entregada en cada etapa en función de la productividad del equipo de desarrollo. La metodología propuesta requiere que la persona

responsable de un producto, ya sea una clase, un caso de uso o un documento de administración de la configuración, realice la estimación. Esto va en contra de las metodologías tradicionales en las que es el líder de proyecto el encargado de realizar todas las estimaciones del proyecto. En los procesos ágiles, al reconocer que las personas son lo más importante del desarrollo las mismas adquieren más responsabilidad en relación a los productos que entregan, debiendo estimar la duración del desarrollo en cada iteración. Esto no quiere decir que los Programadores estarán a cargo de hacer la estimación del proyecto completo - el Líder será el que lo realice consultando a todos los miembros del equipo, ya que este es responsable por la concreción del proyecto entero - pero si serán responsables por los productos que construyen durante las tareas que realizan todos los días.

La metodología propuesta fomenta el concepto de estimación descentralizada. Cada persona estimará el fruto de sus tareas por la técnica que considere necesaria. Una consecuencia de esto es que las personas no sienten que se les ha impuesto un cronograma al que deben atenerse pudiendo ser reprendidas en caso de retrasos. El responsable de un producto es la persona más adecuada para medir su ritmo y estimar los tiempos necesarios. Al principio esto podrá resultar difícil para aquellas personas acostumbradas a tener fechas y tareas impuestas desde arriba, pero ahí es donde entra en juego el Coordinador para capacitar a los miembros del equipo y perfeccionarlos en las estimaciones futuras.

La estimación para una iteración dada será realizada durante la iteración previa, de forma de tener la planificación aprobada por el cliente previo a embarcar en las actividades correspondientes al equipo en el desarrollo. Esta estimación será efectuada por todo el equipo junto al cliente y en la misma será validada la funcionalidad a ser próximamente construida.

4.2.6 ENFOQUE EN LA ARQUITECTURA

Una de las ideas propuestas por la metodología propuesta consiste en una temprana definición de la arquitectura del sistema. La arquitectura no es un concepto que pueda ser explicado en un par de palabras. Consiste más

bien en una suma de aspectos que están relacionados con el diseño de un sistema.

Ya sea que estemos construyendo un edificio, una prensa hidráulica o un software empaquetado, debemos tener algún elemento de alto nivel que refleje las decisiones que se van tomando en relación al diseño de la construcción. Haciendo un paralelismo con el paradigma de objetos, considerando al sistema como un paquete a ser embebido en un ambiente específico, la arquitectura estaría dada por:

- La organización del mismo, definida por los paquetes o clases que contiene y las relaciones entre los mismos.
- Las interfaces que provee al mundo exterior.
- Los paquetes que requiere para su funcionamiento.
- Los servicios que provee, realizados a partir de métodos en clases, que deben dar valor a algún actor que interactúa con el sistema.
- Los patrones, estándares, frameworks utilizados para su construcción.

Cabe destacar, que la arquitectura mantiene un cierto nivel de abstracción dentro de la disciplina de diseño. No contempla los detalles de más bajo nivel, como la implementación interna de las clases y métodos, ni analiza en detalle los protocolos a ser utilizados por los sistemas. Simplemente propone un conjunto de vistas [Kruchten, 2000] que enfatizan distintos aspectos del software. Los mismos tienen que ver con:

- Organización lógica
- Funcionalidad
- Concurrencia
- Distribución del software en la plataforma

Gracias a la arquitectura logramos la integridad conceptual necesaria para dirigir nuestro proceso. La importancia de esta abstracción hace necesaria la definición de un rol que participe activamente en su creación y que ya ha sido detallado con anterioridad, el Arquitecto.

4.2.7 ARQUITECTURA ÁGIL

A continuación se detalla cuál es el propósito de la arquitectura en el marco del proceso ágil propuesto, cómo se construye y cómo se mantiene.

En este punto, y a diferencia de la mayor parte de las metodologías ágiles, la metodología propuesta propone definir una arquitectura candidata en forma temprana. La idea subyacente de este patrón deriva de analizar la curva del costo de cambio propuesta por Barry Boehm[27] originalmente. En la misma, [Boehm et al 2000] indicaba que el costo de reparar un defecto descubierto durante la fase de requerimientos resultaba 200 veces más barato que reparar el mismo defecto en la fase de mantenimiento. Esto está graficado en la Figura 4.13, que muestra cómo se obtienen ahorros en una relación de hasta 200:1 al encontrar errores en fases tempranas versus a encontrarlos en fases más tardías. Este análisis llevó a [Boehm, 1988] a proponer el Modelo en Espiral, que mitigaba en forma temprana los riesgos de introducir defectos en la fase de requerimientos mediante la realización de sucesivas iteraciones de validación con el cliente.

Sin embargo, detrás de esta premisa estaba la idea de tener un entendimiento global de todos los aspectos funcionales de la aplicación, creando documentos de especificación completos, detallados, y con todas las normas de calidad requeridas. Esto terminaba degenerando el concepto formulado por el modelo en cascada que una vez que los requerimientos estaban bien definidos la construcción proseguía como una secuencia de fases. Ya han sido descritas todas las falencias de este modelo, por lo cual ahora se analizará como sería la curva del cambio en el universo de las metodologías ágiles.

Esta misma noción aplicaba a la idea de realizar cambios a los requerimientos. Dadas las características humanas que hacen muy difícil las actividades de relevamiento debido a que el cliente va cambiando aprendiendo lo que desea con el tiempo y derivando de la Figura 4.13 ya

---

[27] Barry W. Boehm es un ingeniero informático estadounidense y también es profesor emérito de esta materia en el departamento de ciencias tecnológicas en la Universidad del Sur de California.

mencionada, se tomaba como referencia en las metodologías tradicionales la Figura 4.14 que mostraba el impacto del cambio en cada fase.

De hecho era el mismo modelo en cascada o secuencial el que contribuía a que todos los cambios siguieran esa curva dado su escasa tolerancia al cambio. Esto fomentaba nuevamente la idea de que realizando una exhaustiva fase inicial de requerimientos el sistema tendría un mínimo nivel de cambios. Este supuesto era una falacia como muestra la evidencia propuesta por Capers Jones[28] presentada en la Figura 4.15. Según este análisis el cambio a los requerimientos en proyectos de software va desde un 10% sobre el alcance total en los proyectos más chicos, hasta un 35% o más en los proyectos más grandes. Queda demostrado que la construcción de software se halla en un dominio de alto cambio y el proceso en cuestión debe tomar esto en cuenta.

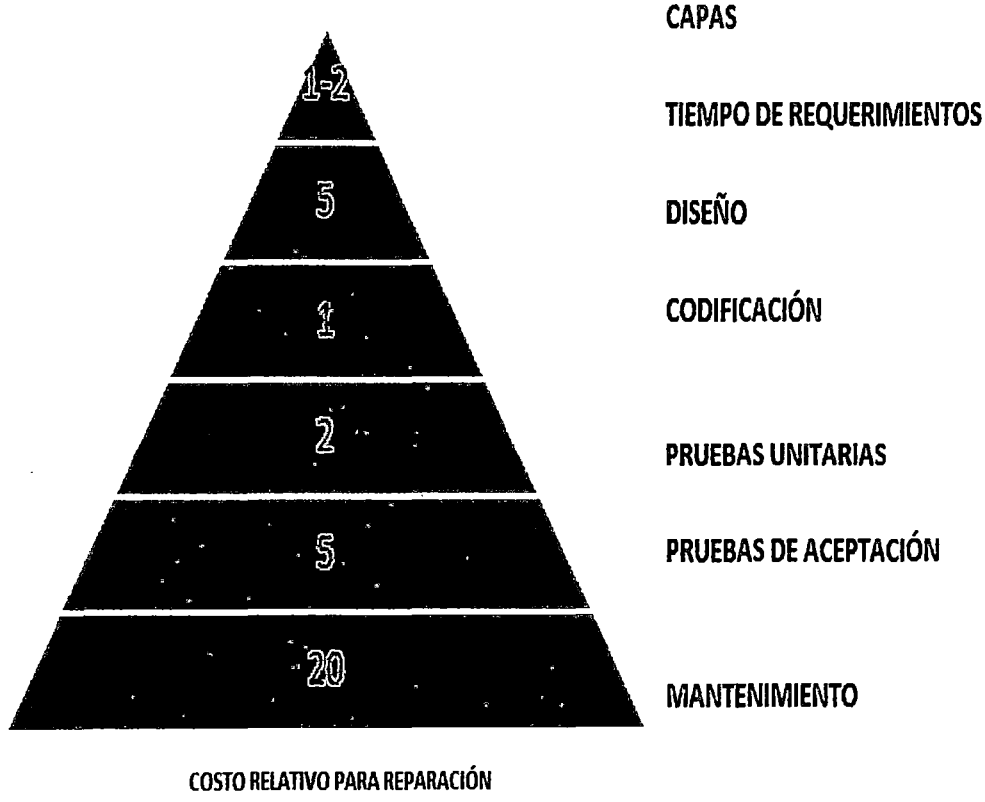


Figura 4.13: El costo relativo de reparar un defecto en diferentes fases del ciclo de vida. Tomada de [Leffingwell, 2001].

[28] Capers Jones es un especialista estadounidense en materia de metodologías de ingeniería de software, y se asocia a menudo con el modelo de punto de función de la estimación de costos.

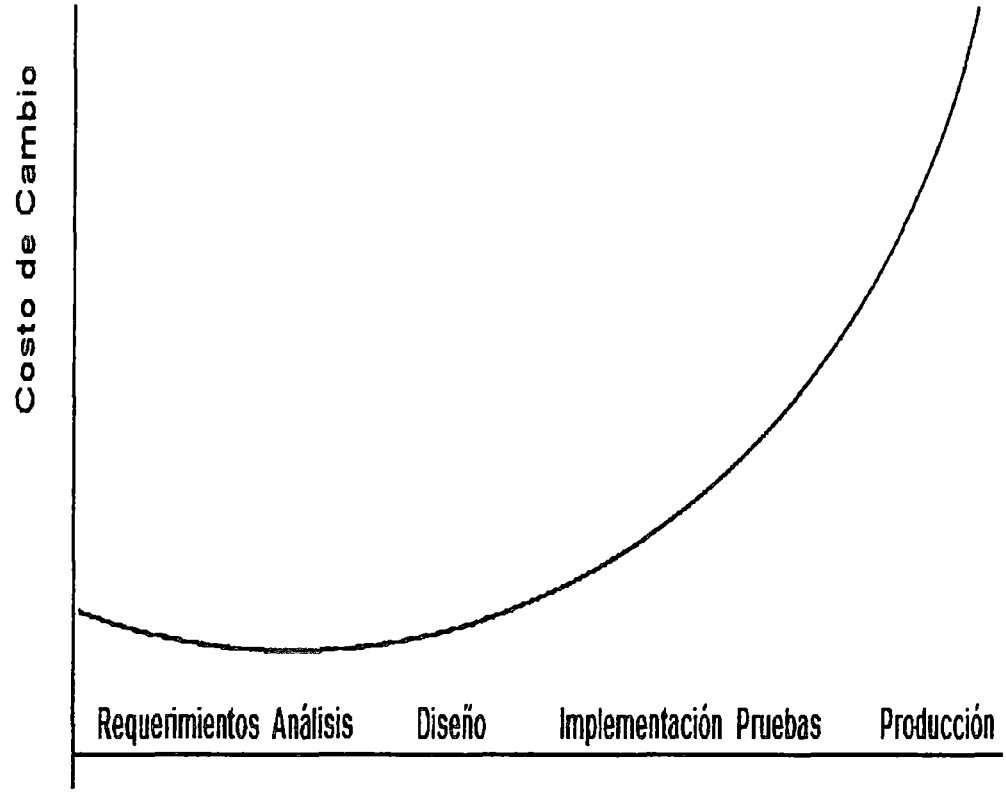


Figura 4.14: Curva del costo de cambio a los requerimientos en diferentes fases del ciclo de vida. Tomada de [Beck, 2000].

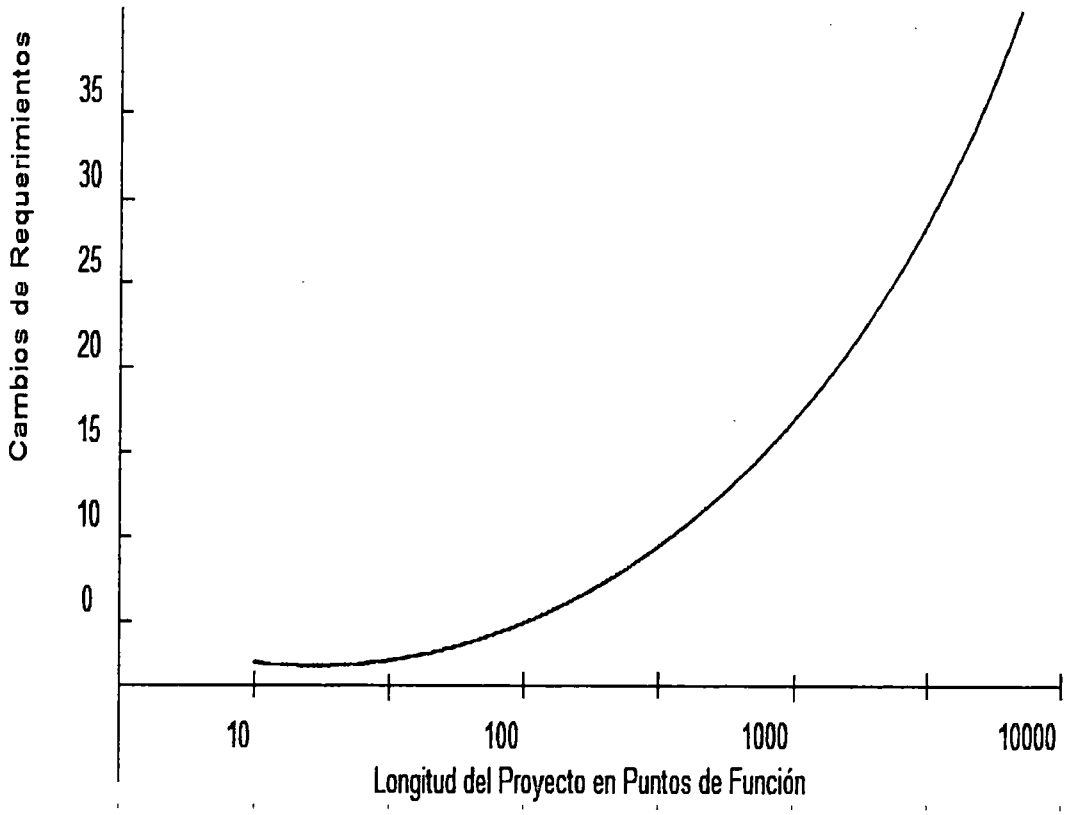


Figura 4.15: Tasas de cambio a los requerimientos en proyectos de software respecto al tamaño funcional de los mismos. Tomada de [Larman, 2003].

La premisa de la metodología propuesta consiste en tener un proceso ágil que tenga alta tolerancia al cambio y que permita que la curva del costo de cambio propuesta por [Boehm, 2000] no escale exponencialmente en la mayoría de los casos. La metodología propuesta sugiere que en los proyectos se tenga una gráfica como muestra la Figura 4.16. En la figura hay dos curvas: la número 1 que crece exponencialmente al igual que la original de Boehm y en la que se manifiestan como la arquitectura candidata del sistema, mientras que en la curva número 2 que es bastante achatada pondríamos todos los cambios relacionados a la funcionalidad del sistema los cuales pueden ser absorbidos en las distintas iteraciones mediante el feedback continuo con el cliente.

La idea de Metodología propuesta es tratar de mover todos los cambios a la curva número 2 [idealmente ese el postulado de XP, que plantea que la curva del costo de cualquier cambio es la curva número 2], pero reconoce

que existen distintas aspectos que implican un alto costo al cambio una vez que se avanza en el desarrollo. Para aquellos factores que revisten las características de la curva número 1, la metodología propuesta sugiere una definición previa al inicio del desarrollo en gran escala y establece al Documento de Arquitectura como el repositorio de estas cuestiones.

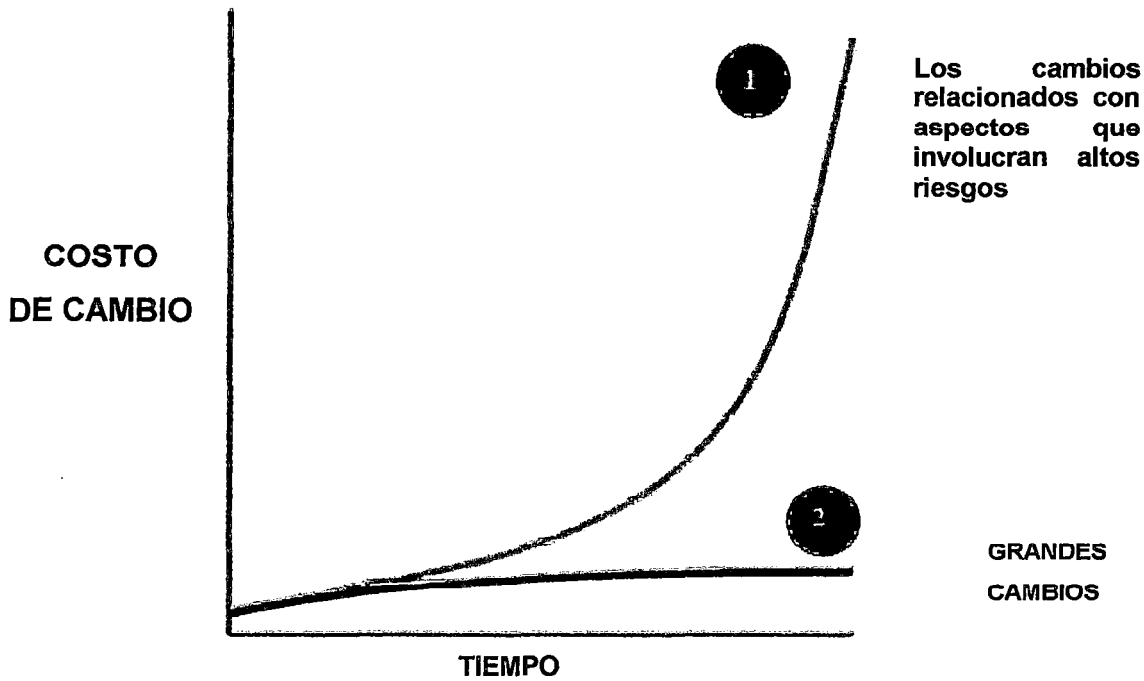


Figura 4.16: El costo del cambio actualizado para una metodología ágil. Tomada de [Poppendieck, 2003]

Sin embargo, dadas las características del proceso resulta burocrático desarrollar un documento de grandes proporciones que guíe el proceso con las decisiones más importantes del diseño, que incorpore extractos de los diversos aspectos que son reconocidos como esenciales para el desarrollo y que sea actualizado en cada iteración para servir a su propósito.

Pero antes de esto se analiza las implicancias del principio de la comunicación en el alcance de la metodología. Siendo que el propósito de la arquitectura es la comunicación de aspectos esenciales del software entre las partes involucradas o stakeholders, ¿no se ve reducido drásticamente su alcance dada la estrecha comunicación propuesta por la metodología?. En

otras palabras, al combinar el principio de Máxima Comunicación con el de Arquitectura Ágil, debemos cambiar la definición de arquitectura tradicional ya que la misma no coincide con las prácticas propuestas.

El principio de Arquitectura Ágil en la metodología propuesta consiste en mantener una visión única, formulada por el equipo de proyecto de las decisiones consideradas críticas para el desarrollo. Será el mismo equipo el encargado de construir, mantener y obtener los beneficios de la misma. En consecuencia, la arquitectura no va a ser un documento estático con una estructura determinada a la cual el equipo se deba ajustar aunque no contribuya al desarrollo. Irá evolucionando con el desarrollo de los componentes y será mantenida hasta el punto que contribuya a la comunicación de los aspectos tecnológicos dentro del grupo. Una vez que el conocimiento esté en las mentes de todos, la misma habrá cumplido su función y el documento podrá ser dejado a un lado. Esto es así debido al principio de Proceso Orientado a las Personas, el cual describe la necesidad de construir aquellos artefactos indispensables para el proyecto de software, aquellos que sirvan algún propósito claro.

Al momento de definir la arquitectura, la metodología propuesta propone una serie de modelos extraídos y extendidos a través de estereotipos del *Unified Modeling Language* [UML]. Los mismos sirven como guía según el tipo de aplicación que se este construyendo. En general, nos referiremos a aplicaciones que se encuentren dentro del espectro de aplicabilidad de la metodología propuesta; es decir, aplicaciones del sector IS/IT, de una escala media o pequeña, con alto grado de involucramiento de los usuarios.

Estos modelos están especificados UML y sirven para describir distintas vistas de la arquitectura necesarias para la comunicación dentro del equipo. Entre los modelos que el arquitecto debería manejar al momento de definir la arquitectura encontramos:

4.2.7.1 DIAGRAMA DE DESPLIEGUE

Definición: el diagrama de despliegue muestra cómo y dónde el sistema será desplegado. En el mismo se muestran nodos físicos, procesadores, y los componentes que corren en cada uno. Como se observa en las figuras este diagrama puede tener distintas implementaciones de acuerdo al grado de detalle que se quiera llegar. Sin embargo, es muy recomendable ya que muestra como la aplicación será llevada al ambiente de producción mediante la separación de los componentes en nodos. Ver figura 4.17 y 4.18.

4.2.7.2 DIAGRAMA DE COMPONENTES

Definición: el diagrama de componentes muestra las piezas de software o entidades que conforman al sistema; en general, un componentes es de más alto nivel que una clase y suele ser implementado en tiempo de ejecución por un número de clases. Ver figura 4.19.

4.2.7.3 DIAGRAMA DE SECUENCIA

Definición: el diagrama de secuencia es una representación de las interacciones entre clases y objetos para llevar a cabo una operatoria a lo largo del tiempo. Se utiliza para mostrar flujos de trabajo, pasaje de mensajes y la cooperación necesaria para brindar un determinado resultado. Ver figura 4.20.

4.2.7.4 DIAGRAMA DE ESTADO

Definición: el diagrama de estado sirve para mostrar como un elemento [clase] cambia de estado a lo largo del tiempo y las transiciones que le son permitidas en cada caso con las correspondientes condiciones. Ver figura 4.21.

4.2.7.5 DIAGRAMA DE CLASES

Definición: el diagrama de clases captura la estructura lógica del sistema – las clases y/o entidades que componen al modelo. Es considerado

un modelo estático ya que muestra las clases existentes y los atributos, métodos de cada una. Ver figura 4.22.

4.2.7.6 DIAGRAMA DE DATOS

Definición: el diagrama de datos sirve para mostrar como el modelo será persistido en un Sistema de Gestión de Base de Datos Relacionales. El mismo está dentro del perfil de UML para modelado de datos y maneja estereotipos como Tabla, Primary Key, Foreign Key, etc. Ver figura 4.23.

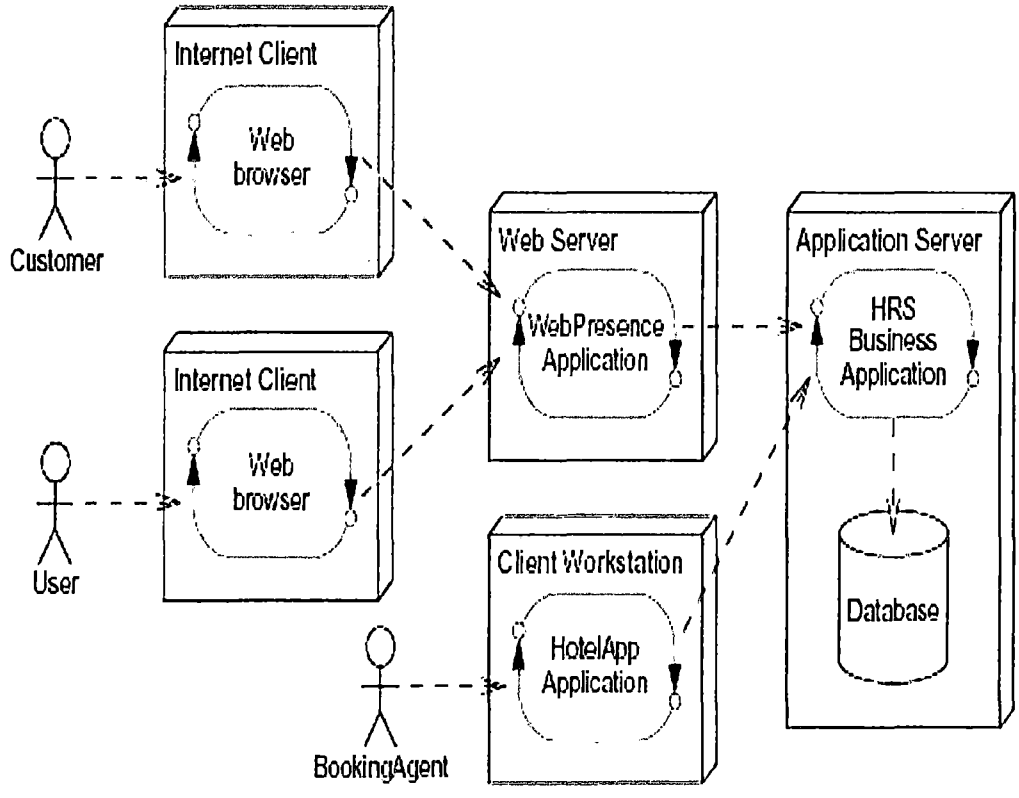


Figura 4.17: Diagrama de Despliegue de alto nivel en UML. Tomada de [Sun, 2003]

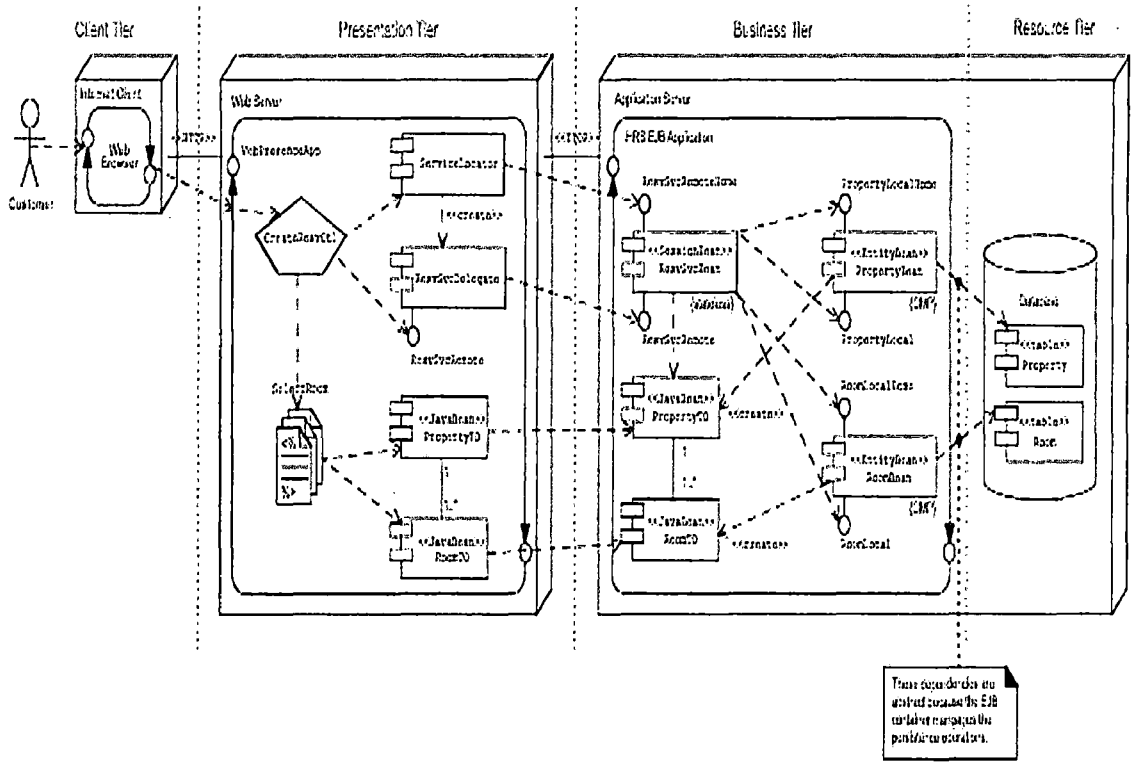


Figura 4.18: Diagrama de Despliegue detallado en UML. Tomada de [Sun, 2003]

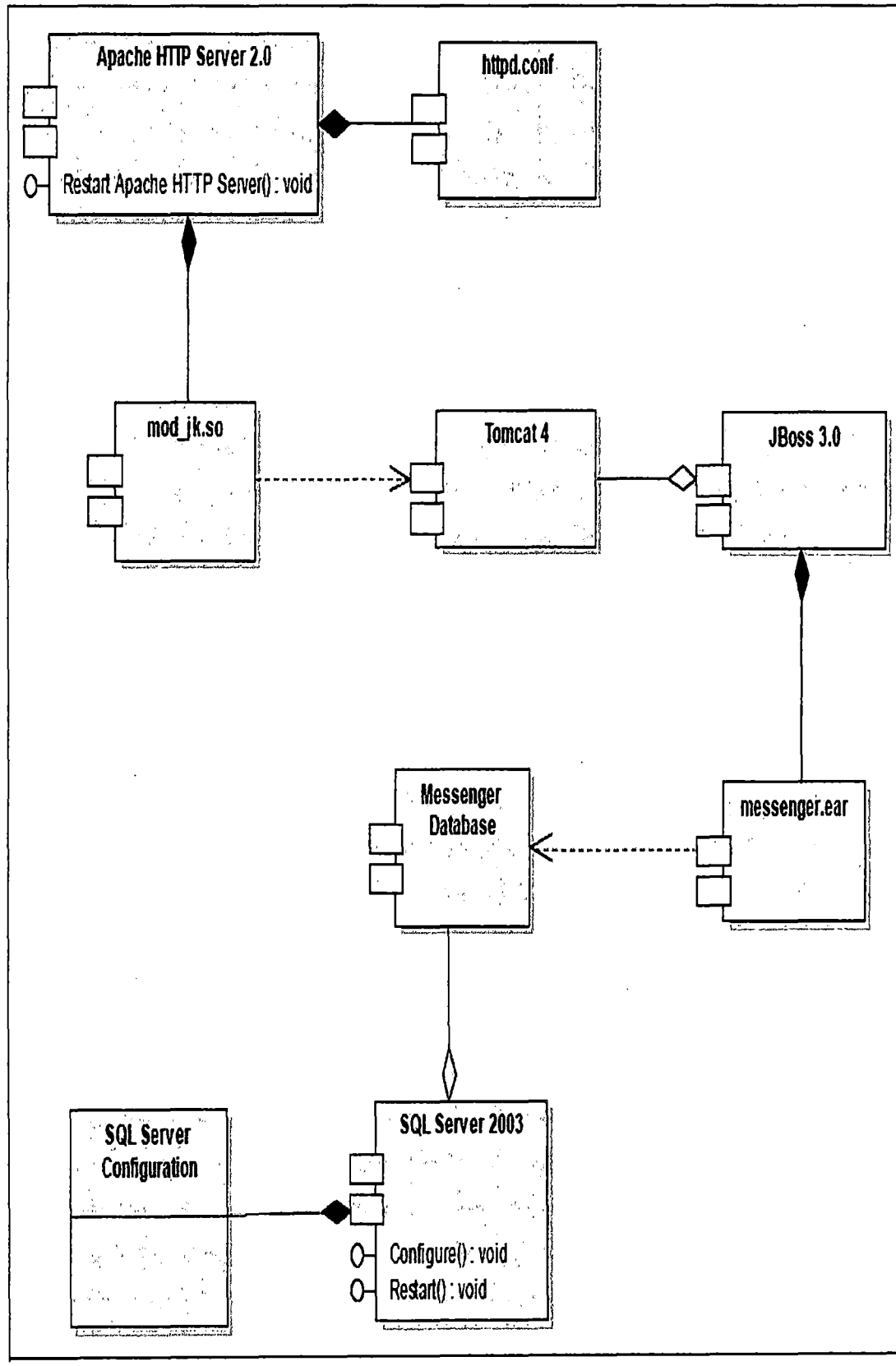


Figura 4.19: Diagrama de Componentes en UML. Tomada de [EA, 2003]

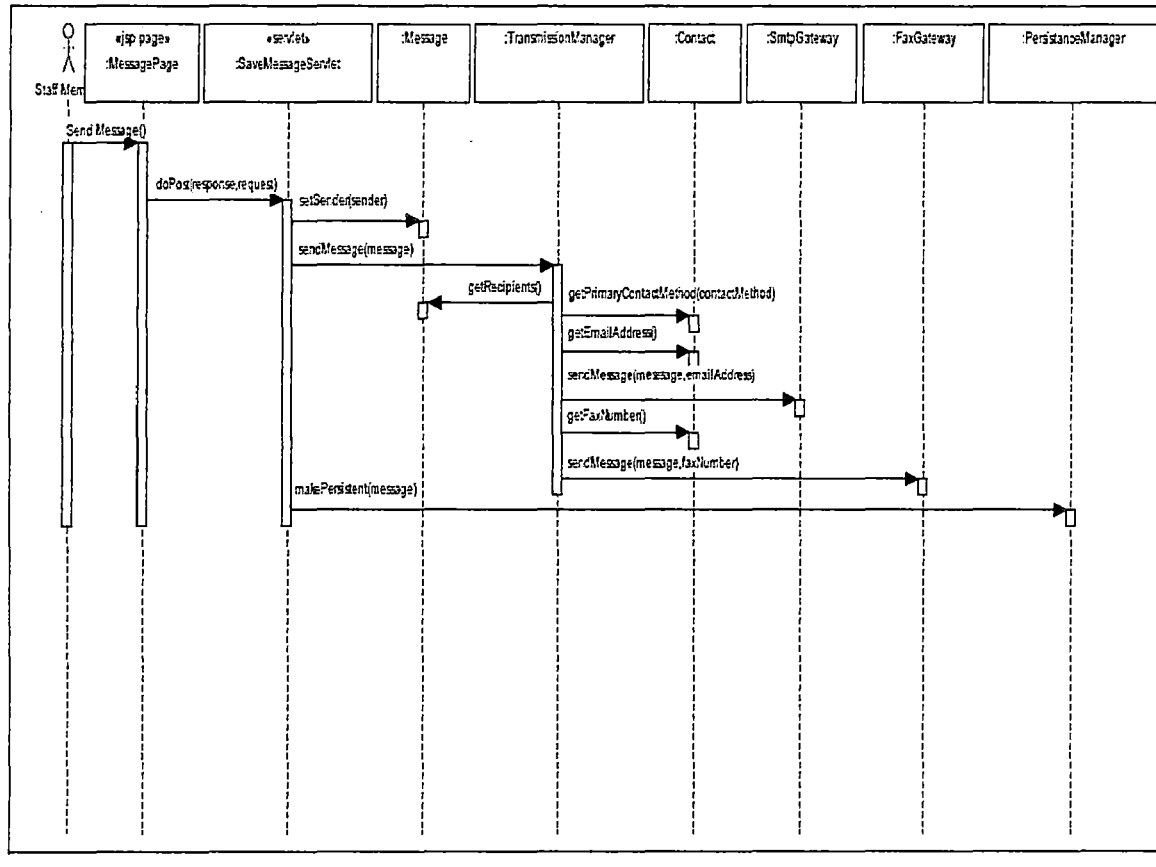


Figura 4.20: Diagrama de Secuencia en UML. Tomada de [EA, 2003].

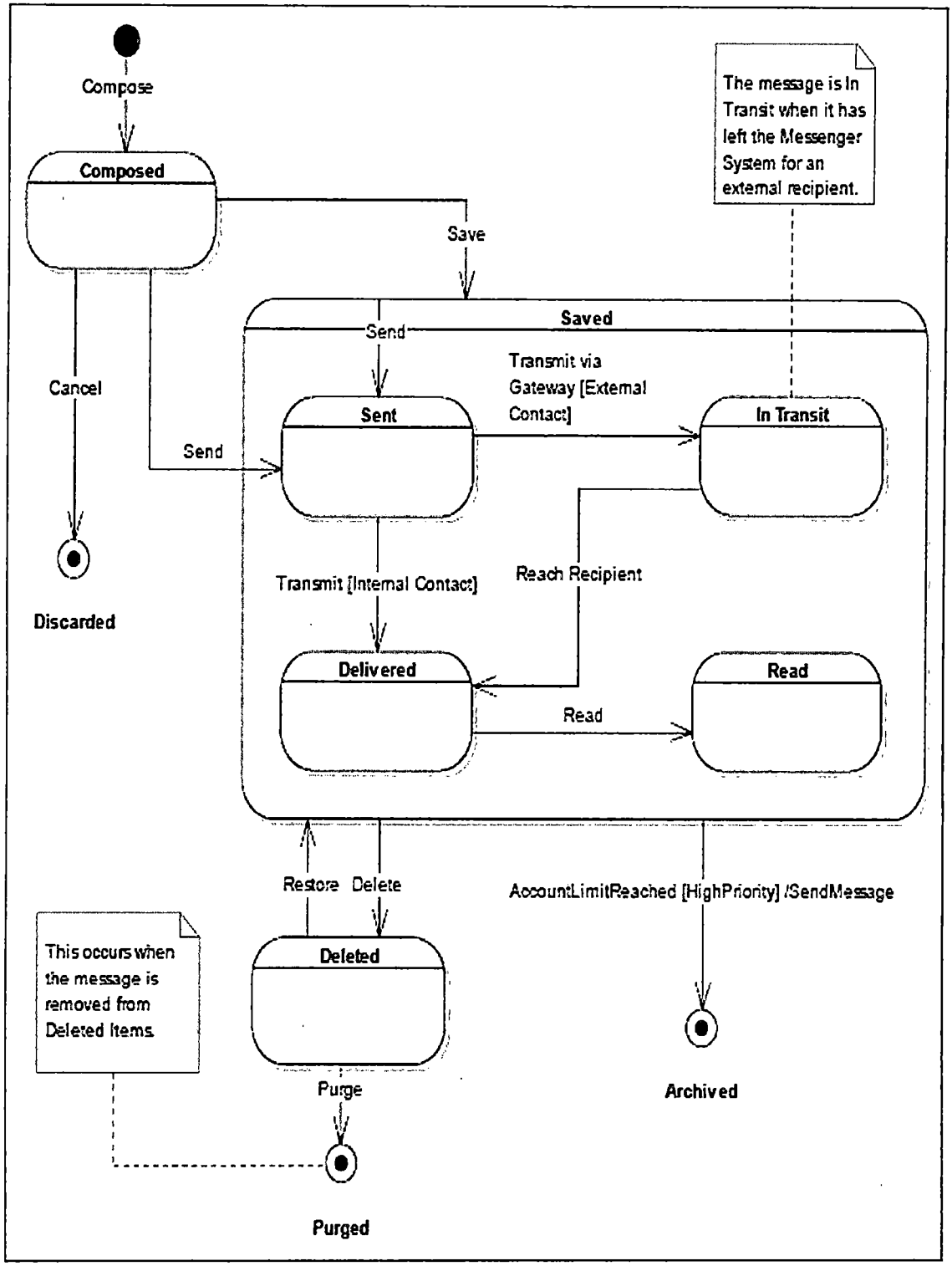


Figura 4.21: Diagrama de Estado en UML. Tomada de [EA, 2003].

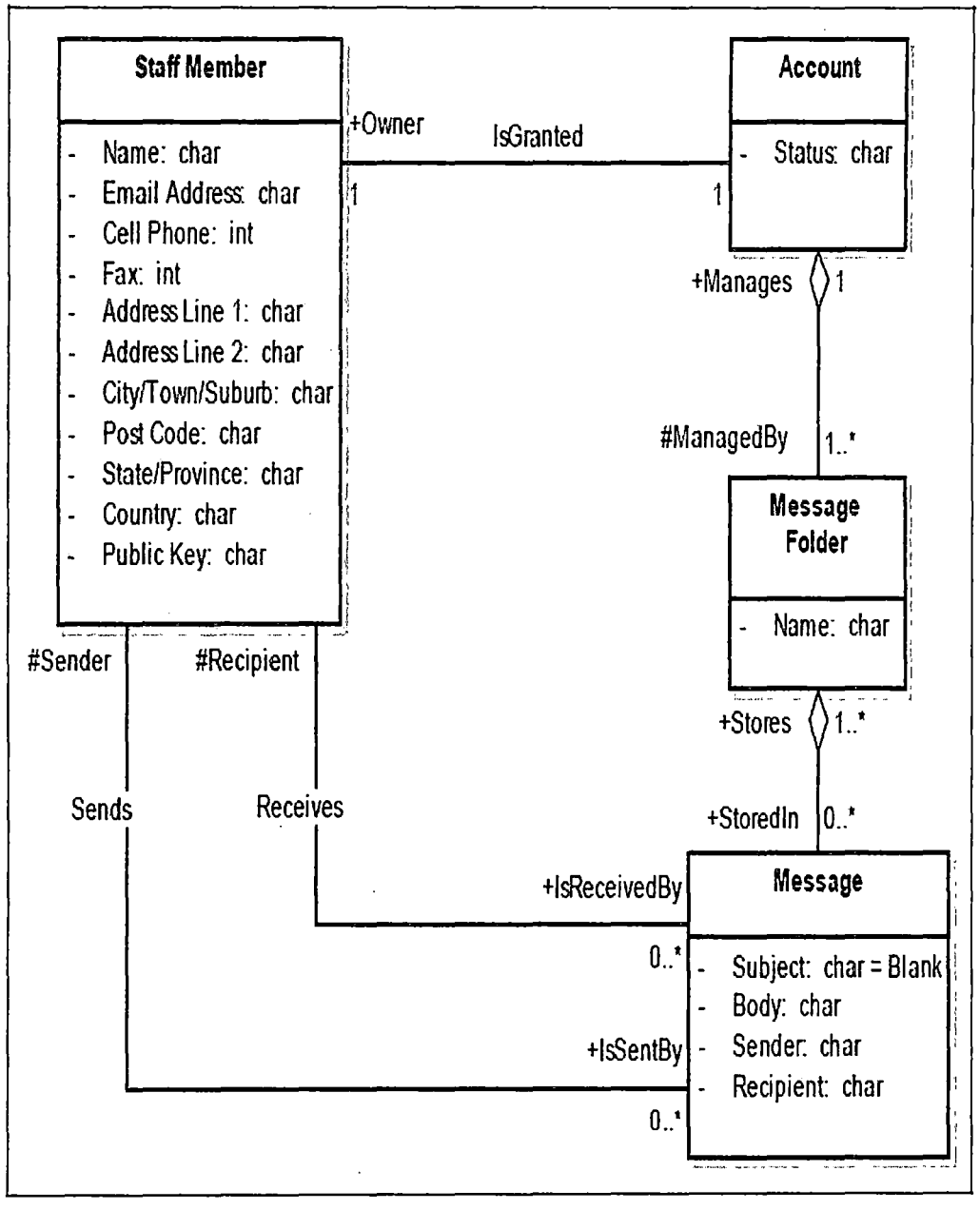


Figura 4.22: Diagrama de Clases en UML. Tomada de [EA, 2003]

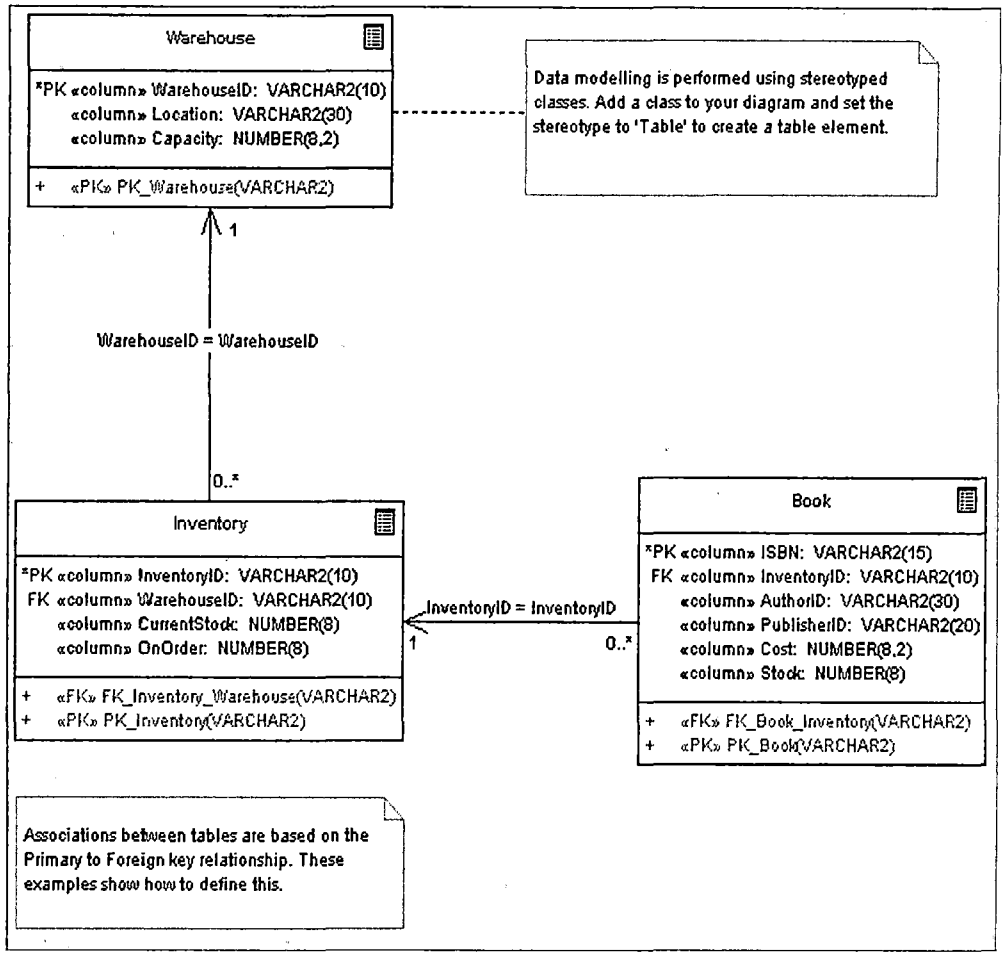


Figura 4.23: Diagrama de Datos en UML. Tomada de [EA, 2003]

Se debe tener en cuenta que debe existir un balance entre el esfuerzo en realizar estos diagramas y la posibilidad de comunicar dichas ideas en forma directa. De acuerdo al patrón de Máxima Comunicación conviene que estos diagramas estén en algún radiador de información ya sea mediante impresiones en hojas grandes colgadas, en una intranet de administración de conocimiento, o simplemente dibujados en pizarrones en los casos que se desee más informalidad. Una vez que el propósito comunicacional es servido no tiene sentido seguir detallando y/o manteniendo estos diagramas.

4.2.8 INTEGRACIÓN CONTINUA

Indudablemente una de las prácticas que hoy se encuentra recomendada por la mayor parte del espectro de metodologías modernas [no solo ágiles] es el de Integración Continua.

Esta práctica nace a partir de los fracasos acarreados por las fases de integración de los modelos tradicionales en que una vez que se tenían todos los módulos construidos se iniciaba una intensa y compleja fase en que los mismos se integraban unos con otros hasta lograr el producto a ser entregado. Era en dicho lapso en que ocurrían los mayores descalabros en materia planificación, excediéndose de manera desmedida el tiempo, haciendo que fracasen proyectos que habían concluido con éxito sus fases anteriores. Una de las causas principales consistía en las complejas interacciones existentes entre los distintos módulos, construidos por distintos equipos, los cuales al ser integrados por vez primera presentaban errores que eran muy difíciles de encontrar por estar generados por distintas personas.

La industria del software aprendió su lección, y ya desde hace algún tiempo, la Integración Continua es considerada esencial en cualquier proceso moderno de desarrollo por más burocrático o minimalista que resulte. Por dar un ejemplo, la metodología Extreme Programming contiene la Integración Continua entre sus 12 prácticas claves. Más aún es de público conocimiento que desde hace muchos años Microsoft viene utilizando esta práctica en su proceso de desarrollo de donde surgió la noción del "daily build", en que todas las noches se integraba, compilaba y linkeaba el código generado para ser luego probado por los casos de prueba automatizados. Steve McConnell [McConnell, 1996] fue uno de los que más analizó esta técnica recomendándola entre sus conocidas *best practices*. Martin Fowler [Fowler, 2002] escribió un artículo bastante descriptivo de los beneficios de esta práctica.

Existen en la actualidad herramientas muy potentes – en algunos casos, gratuitas por ser de código abierto – las cuales permiten que esta

práctica pueda ser llevada a la realidad con un mínimo esfuerzo. Un ejemplo de herramienta Open Source de estas características es el Cruise Control que funciona en conjunto con Ant, CVS, y JUnit para llevar a cabo los builds continuos en una máquina.

Sin embargo, se debe recordar que hasta este instante estamos verificando la sintaxis y semántica del código, pero no sabemos nada respecto a si cumple las necesidades del usuario. Para esto se tienen las pruebas funcionales automatizadas o manuales que deberá crear o especificar el analista de pruebas junto con el cliente.

4.2.9 ENFOCADO A LA PERSONA Y NO AL PROCESO

Una de las cuestiones en que se enfoca la metodología propuesta es la persona. Este término refiere a las personas involucradas en el desarrollo de software y a todos los aspectos relacionados con las mismas.

Las primeras nociones de *Peopleware*[29] surgieron alrededor de 1969, cuando la industria del software apenas comenzaba a madurar, de manos de Gerald Weinberg en su libro The *Psychology of Computer Programming*, posteriormente reeditado en 1998 [Weinberg, 1998]. En el mismo, se relativiza los aspectos relacionados con el proceso y las técnicas utilizadas para construir software abocando la importancia que tenía el factor humano en el mismo. La comunidad informática no incorporó las consideraciones de Weinberg, quien siguió escribiendo acerca de estos temas durante el tiempo subsiguiente. Pasaron algo más de 15 años hasta que Tom DeMarco tomara nuevamente estas ideas para su libro Peopleware: *Productive Projects and Teams* [DeMarco, 1987] también reeditado en 1999. El libro de DeMarco tuvo mayores repercusiones en la comunidad que las que había tenido su predecesor, aunque no logró imponer el cambio necesario para considerar

[29] Peopleware [del inglés, people, gente y ware, mercancía] es un término utilizado para designar uno de los tres aspectos centrales de la tecnología de computadores, siendo los otros dos: hardware, software. Peopleware puede referirse a cualquier cosa que tenga que ver con el papel de las personas en el desarrollo o uso de software y sistemas hardware, incluyendo cuestiones como productividad de los desarrolladores, trabajo en equipo, dinámicas de grupo, la psicología de la programación, gestión de proyectos, factores de organización, diseños de interfaces de usuario y interacción hombre-máquina.

las cuestiones humanas como dominantes en el desarrollo del software. Cabe destacar que algunas empresas sí tomaron las prácticas de DeMarco con algunos resultados bastante exitosos – el ejemplo más destacable es Microsoft.

Llevando las ideas de todas estas fuentes al contexto de esta tesis, la metodología propuesta toma a las personas como partes esenciales del desarrollo y las encuadra en un marco humano necesario para el entendimiento del impacto que las mismas tienen en la construcción de los diversos artefactos. Dicho marco humano se centra en tres áreas de estudio: organización, área de desarrollo e individuo.

Dentro de la organización agrupamos aquellas cuestiones externas al área de desarrollo que no son controladas por el mismo.

- El lugar de trabajo.
- Recursos de tecnología disponibles.
- La cultura de la empresa.

Por área de desarrollo entendemos la forma en que se relacionan y organizan los individuos para realizar el desarrollo.

Dentro del análisis del individuo se encuentran los factores que afectan a un individuo al momento de desarrollar software. Las personas necesitan tener un propósito para lo que hacen y esto toma mayor importancia en relación a su trabajo ya que éste ocupa en promedio el 35% del tiempo durante el que realizan actividades. Consecuentemente, se debe fomentar que el individuo esté cómodo en su trabajo y que exista una buena relación con sus pares. Para esto es de suma importancia que el Líder de Proyecto monitoree las dinámicas que entran en juego en el grupo así como el comportamiento de los individuos.

La cultura de una empresa se define como el conjunto de ideologías, valores, metas, que perciben los individuos que conforman la misma. Representa el folclore de las personas que interactúan bajo una misma organización. Una de las primeras cuestiones con las que debe inmiscuirse rápidamente un individuo al comenzar a formar parte de una empresa es su

cultura. Gracias a ésta el empleado se irá adaptando, cambiando sus valores, formulando nuevos juicios y terminará poniéndose las mismas metas que el resto de las personas o abandonará el intento por no sentirse cómodo con la misma. La cultura de la empresa tiene tanta relevancia como el trabajo en sí para el que fue contratado el empleado.

Para analizar cómo impacta este factor en el desarrollo de software, utilizando la metodología propuesta, deberemos comenzar por analizar la cultura de la empresa en que vamos a implementar el proceso. La empresa ideal para el éxito de un proceso ágil es aquella que comparte un principio de ***compartir información*** versus ***mantener información***. Entendemos por compartir información la aptitud de las personas unidas bajo un mismo objetivo de colaborar para que el conocimiento sea transferido de donde está contenido a donde es requerido. Esta noción se contrasta con la de mantener información en la que un grupo de personas compite por acumular conocimiento y se esfuerza en no comunicarlo a quienes lo necesiten.

Estos extremos de una recta representan el grado de madurez de la disciplina de Administración de Conocimiento [KM] dentro de la organización. Se entiende por KM al proceso organizacional para retener, organizar, compartir, y actualizar conocimiento crítico para la performance individual y la competitividad organizacional [Davenport, 1998].

4.2.10 CALIDAD EN EL PROCESO

El referirnos a la metodología propuesta como una metodología ágil no quiere decir que el énfasis puesto en los entregables y la rapidez del desarrollo no tengan en cuenta el problema de la calidad.

Tomando como marco de referencia al CMM, es sabido que la mayoría de las empresas del país se encuentran en el nivel 1, denominado Inicial. Una de las razones por las cuales esto ocurre radica en la falta de un proceso de desarrollo maduro y en la tendencia inherente en la gente de sistemas de seguir el modelo Codificar y Probar. Esta tendencia se debería modificar de forma de seguir un proceso relativamente disciplinado, generando documentación de análisis/diseño previamente, haciendo la

prueba luego de la codificación, siguiendo entregas definidas. Sin embargo, al momento en que el tiempo apremia se dejan de lado todas estas prácticas, desapareciendo el proceso al punto de volver al caos del que se deseaba escapar.

Una de las razones de este fracaso consiste en la falta de convencimiento pleno por parte del equipo de desarrollo en el proceso que utilizan diariamente. Mientras los hitos se vayan cumpliendo siendo las entregas aceptadas por los clientes el equipo se siente confidente de que su proceso los llevará al éxito y que además lo están siguiendo correctamente. Todos comentarán la necesidad de mantener un proceso disciplinado mostrando las experiencias pasadas como demostración de los resultados y de los productos entregados.

Podrá llegar el momento en que el proyecto comience a retrasarse en las entregas, a disminuir la calidad o a remover funcionalidad de las mismas. Será en estos momentos en que nacerá un sentimiento de desesperación que comenzará en el management descendiendo en la jerarquía hasta el equipo de desarrollo. Se pedirán entregas imposibles, las cuales llevarán a los programadores a codificar en forma masiva, dejando de lado completamente cualquier viso de proceso, erradicando completamente la calidad del software construido.

Por estas cuestiones es importante que el proceso se tenga en alta estima y que la gente esté convencida de su utilización. En caso contrario, el proceso no cumple su función. Para mantener la salubridad del mismo incluimos en la próxima sección una serie de disciplinas que permiten tener siempre un proceso de calidad que se adecúe a las necesidades de la gente que lo utiliza.

4.3 APORTES DE LA METODOLOGÍA AL ESPECTRO METODOLÓGICO

A continuación se exponen los aportes que realiza la metodología propuesta al universo de las metodologías ágiles, en base a un análisis exhaustivo de los procesos existentes y de la identificación de aquellos aportes que hacen de la metodología propuesta un proceso ágil único.

4.3.1 NOMENCLATURA ESTANDARIZADA

Una de las primeras facilidades que una persona se encuentra al aprender de la metodología propuesta es el uso de nombres estandarizados para todos sus elementos. Muchos de los nombres utilizados han sido tomados del RUP [RUP, 2002] que sirve como base de conocimiento para procesos de desarrollo. También existieron fuertes influencias del SWEBOK [SWEBOK, 2001] que comprende el cuerpo de conocimiento de la ingeniería de software y que es fiel esperanza del autor que dicho proyecto siente las bases hacia una profesionalización de nuestra disciplina. Gracias a esta estandarización, la metodología propuesta contribuye a un global entendimiento de los conceptos involucrados lo cual habilita una comunicación más fluida entre las personas involucradas.

4.3.2 ADMINISTRACIÓN DE PROCESO

La metodología propuesta propone distintos patrones enmarcados en una disciplina que pueden ser utilizados para adecuar el proceso en distintas etapas. Los patrones sirven para que la metodología propuesta pueda ser implementado exitosamente y para adaptar el mismo a las necesidades particulares del proyecto. Esto logra captar el feedback de la gente que es usuaria del proceso y dinámicamente modificarlo para que cumpla mejor sus objetivos.

4.3.3 ADMINISTRACIÓN DE PERSONAS

La metodología propuesta propone distintos patrones enmarcados en una disciplina que pueden ser utilizados para manejar mejor el factor humano dentro del desarrollo. Los patrones ayudan a mejorar el ambiente de trabajo de los individuos, mejorar la motivación de los grupos de trabajo y mitigar las fricciones que puedan existir. El resultado será una adopción rápida y eficaz de la metodología propuesta, tolerante a la diversidad de personalidades existentes.

4.3.4 ADMINISTRACIÓN DEL CONOCIMIENTO

La metodología propuesta explícitamente establece una disciplina de Administración del Conocimiento que contribuye al aprendizaje organizacional. La misma establece una serie de prácticas ágiles que ayudan a que el conocimiento novedoso generado durante un proyecto sea capturado para una posterior recuperación. El objetivo es que el conocimiento se disemine por los grupos de trabajo y que se vaya aprendiendo de la experiencia para el mejoramiento de las personas y la agilización del proceso.

4.3.5 PROCEDIMIENTO PARA LA IMPLEMENTACIÓN

Existe una frase en el terreno de la Ingeniería de Software que dice: "Lo difícil no es definir una metodología, sino implementarla en la organización". Dada esta realidad, la metodología propuesta plantea una serie de consideraciones, surgidas a partir de las experiencias de otras áreas de conocimiento, que contribuyen a su implementación en un grupo humano.

4.3.6 PRÁCTICAS PROBADAS POR LA EXPERIENCIA

Las prácticas realizadas por la metodología propuesta han sido probadas durante suficiente tiempo como para concederles un status de "mejores prácticas".

En especial, la metodología propuesta reúne un conjunto de prácticas que considera adecuadas a la aplicabilidad de la metodología y que ha sido demostrada por la experiencia propia del autor.

4.3.7 PATRONES DE ARTEFACTOS

La metodología propuesta ofrece una serie de patrones que ayudarán al Coordinador que lleve a cabo la implementación a elaborar los artefactos sugeridos. Estos han sido utilizados en distintos proyectos y plasman de una manera concisa la información que se desea transmitir. La idea de los

mismos es adaptarlos de acuerdo a la realidad de los proyectos manejados por la organización.

4.3.8 BAJA CURVA DE APRENDIZAJE

La metodología propuesta está pensada para ser utilizada tal cual establece esta tesis. No es necesario poseer conocimiento avanzados en conceptos teóricos ni disponer de una estructura organizacional dedicada exclusivamente a la implementación del proceso. Las prácticas han sido elegidas entre otras cosas por adaptarse a la forma de trabajo de la gente ayudando a resaltar aquellos elementos que hacen a una metodología ágil.

Asimismo, las prácticas resultan sencillas de transmitir y de implementar para toda persona que conozca las ideas detrás de los procesos de desarrollos iterativos e incrementales.

CAPÍTULO V

APLICACIÓN DE LA METODOLOGÍA PROPUESTA A UNA ENTIDAD DEL ESTADO

En este capítulo, se presenta la validación del trabajo de investigación que se ha abordado aplicando el método de investigación en acción descrita anteriormente. Se describe el sistema de software que se ha desarrollado como caso de estudio. Así, se ofrece una visión global del método de desarrollo, resaltando el tema relacionado con el Desarrollo de Software.

El Sistema Nacional de Pensiones [SNP] requiere para su servicio de un sistema de información. Debido a los constantes cambios en el sistema, este necesita ser atendida en forma rápida.

La Entidad del Estado encargada de administrar el Sistema Nacional de Pensiones requiere de un proveedor de fábrica de software, que le ayude a mejorar y automatizar de una manera rápida y oportuna los requerimientos Para ello cuenta con oficinas descentralizadas en regiones estratégicas y además de una Oficina de Tecnologías de Información[30] [OTI], el cual vela por manejar y alinear los objetivos organizacionales a la aplicación de los recursos informáticos.

---

[30] Entre sus funciones se encuentran la de ejecutar el mantenimiento de las aplicaciones informáticas, ejecutar el desarrollo e implementación de aplicaciones informáticas, administrar y ejecutar el Plan de Pruebas correspondiente a la puesta en producción de desarrollo y mantenimiento de las aplicaciones, y proponer, ejecutar y supervisar el cumplimiento de los estándares para la administración de los recursos de hardware y software.

5.1 DESCRIPCIÓN DEL SERVICIO

5.1.1 ALCANCE DEL SERVICIO

Realizar las actividades de desarrollo de nuevos sistemas informáticos, el mantenimiento de los sistemas existentes en la Entidad del Estado, con la finalidad de satisfacer las necesidades de eficiencia y automatización de los procesos de la Institución considerando el cumplimiento de los estándares de desarrollo y seguridad de la Entidad del Estado, Disposiciones y Normas sobre Ecoeficiencia[31] de la Entidad del Estado, así como las consideraciones en los Sistemas de Gestión de la Calidad y las Normas de Control Interno de la Entidad del Estado.

5.1.2 CARACTERÍSTICAS DEL SERVICIO

Las características del presente servicio, comprende a los procesos a ser soportados por la fábrica del software las cuales se muestran en el gráfico a continuación [Modelo de fábrica de software]:

[31] Está basado en el concepto de crear más bienes y servicios utilizando menos recursos y creando menos basura y polución.

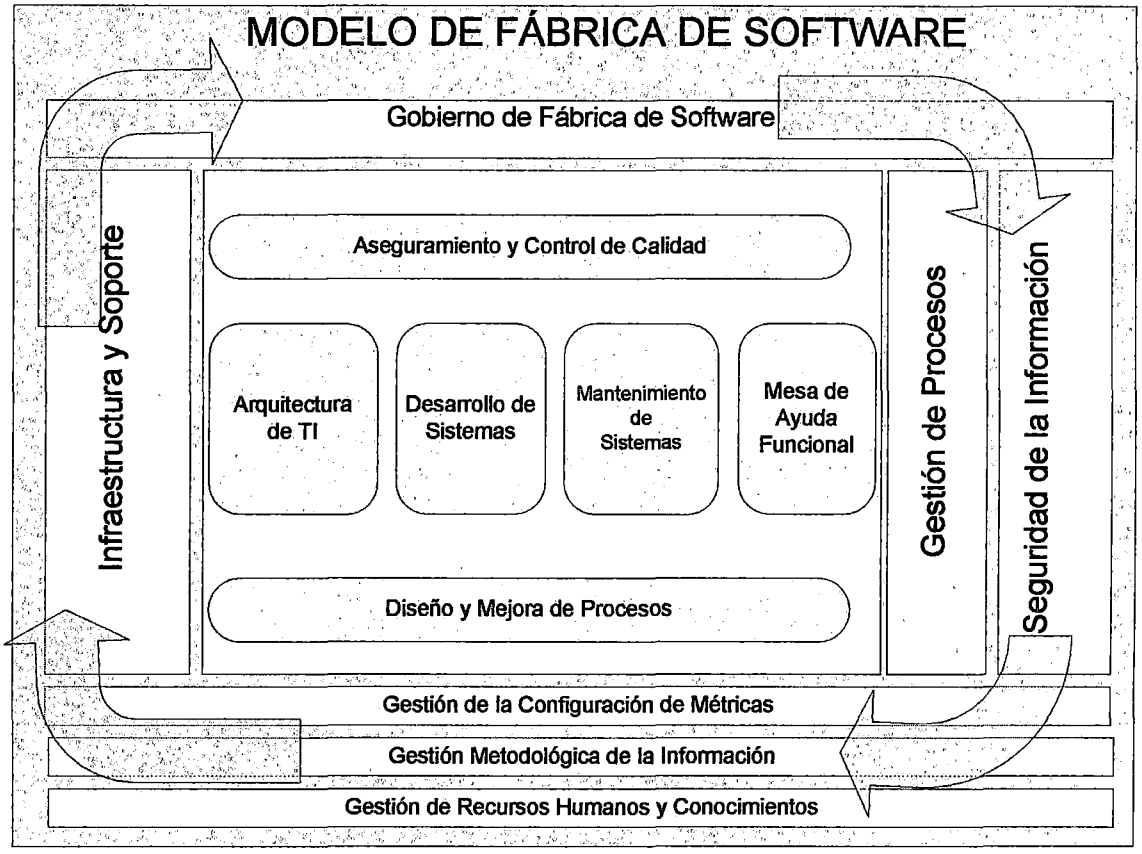


Figura 5.1: Modelo de Fábrica de Software. [Entidad Pública, 2010]

Los procesos principales que la fábrica del software disponga para el presente servicio están relacionados con los proyectos especiales, gestión de la demanda, mantenimiento de sistemas, y aseguramiento y control de calidad, los cuales se detallan a continuación.

5.1.2.1 PROYECTOS ESPECIALES

Esta línea se encarga de la atención de requerimientos de análisis o evaluaciones sobre posibles desarrollos relacionados a los sistemas de información de Entidad del Estado, así como el desarrollo de nuevos módulos, producto de integraciones con otros sistemas o mejora de módulos existentes o migración de información necesaria para un nuevo módulo o sistema, análisis y depuración de datos, así como, rediseño y mejora de procesos que se orientan a la automatización. Finalmente, también pueden surgir requerimientos de estudios y/o evaluaciones técnicas sobre los

sistemas involucrados en el Servicio. En el servicio se pueden presentar los siguientes tipos de proyectos:

- Proyectos de automatización de procesos [desarrollo de sistemas].
- Proyectos de automatización da procesos con SOA[32] [desarrollo de sistemas con SOA].
- Proyectos de rediseño y mejora de procesos [para su futura automatización].

5.1.2.2 GESTIÓN DE LA DEMANDA

Proveer de recursos humanos especializados a fin de apoyar a la Oficina de Tecnología de la Información [OTI] en la Gestión de la Demanda y en la Gestión del Portafolio, para lo cual se contempla por lo menos los siguientes aspectos/ actividades/ entregables:

- Propuesta de metodologías y técnicas para la gestión de la demanda de TI[33], y para la gestión del portafolio de la OTI[34], basado en estándares internacionales, y adaptándose de acuerdo a las características y necesidades de la Entidad del Estado.
- Propuesta de organización para la gestión de la demanda de TI y gestión del Portafolio OTI.
- Evaluación y propuesta de herramientas automatizadas requeridas.
- Aplicación de las metodologías, técnicas y herramientas a fin de gestionar la demanda de TI que llega a la OTI.

---

[32] Arquitectura orientada a servicios de cliente, es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

[33] Tecnologías de Información. Se entiende como "aquellas herramientas y métodos" empleados para recabar, retener, manipular o distribuir información. La tecnología de la información se encuentra generalmente asociada con las computadoras y las tecnologías afines aplicadas a la toma de decisiones [Bologna y Walsh, 1997].

[34] Oficina de Tecnologías de Información. Órgano de apoyo de la entidad del estado, encargada de gestionar, administrar los recursos informáticos.

- La fábrica de software se basa en marcos referenciales o modelos como: COBIT[35], ITIL[36], PMI, CMMI, Val-E Formulación y Evaluación de Proyectos, Análisis Costo-Beneficio y otros que sea conveniente tener en cuenta.
- A nivel de proyecto, se requiere incidir en ciertos aspectos como: Identificación de soluciones alternativas, evaluación de factibilidad operativa y evaluación costo—beneficio y costo-efectividad.
- A nivel de portafolio, se requiere incidir en ciertos aspectos como: criterios de categorización, criterios de selección, métodos de selección, priorización y balanceo.
- Operar el proceso de gestión de demanda, bajo la supervisión de la OTI- Entidad del Estado.
- Operar el proceso de gestión de portafolio, bajo la supervisión de la OTI- Entidad del Estado.
- Incorporar mejoras a las metodologías, técnicas y herramientas, de acuerdo a la forma en que van a ir ejecutando, según lo apruebe Entidad del Estado.
- Capacitación al personal de Entidad del Estado en estos temas.

5.1.2.3 MANTENIMIENTO DE LOS SISTEMAS

Consiste en la atención de requerimientos solicitados por las áreas usuarias autorizadas, a iniciativa del Contratista u otros que se definan. El proceso de mantenimiento del sistema se basa en las siguientes metodologías, modelos y procedimientos:

- Modelo de Gestión de los Ciclos de Producción de Requerimientos de Sistemas.
- Procedimiento de Atención de Requerimientos de Sistemas.

---

[35] COBIT [Control Objectives for Information and related Technology] es el marco aceptado internacionalmente como una buena práctica para el control de la información, TI y los riesgos que conllevan.

[36] Desarrollada a finales de 1980, la Biblioteca de Infraestructura de Tecnologías de la Información [ITIL] se ha convertido en el estándar mundial de de facto en la Gestión de Servicios Informáticos.

- Procedimiento de Elaboración, Revisión y Aprobación de Manuales de Sistemas de Información.
- Metodología de Mantenimiento de Sistemas.
- Metodología de Aseguramiento de la Calidad del Software y del Proceso de Desarrollo y Mantenimiento.
- Método de estimación de tiempos para la atención de requerimientos.
- Procedimiento de Actualización de Información en el Portal Web y Página Institucional de la Entidad del Estado.

Aquellos requerimientos que luego de su pase a producción generen errores imputables al desarrollo y/o mantenimiento realizado por la fábrica de software, deberán ser solucionados por el mismo a costo propio, sin que se vea afectadas las líneas de producción y los tiempos de atención de los requerimientos programados sin perjuicio a las penalidades que la Entidad del Estado pudiera ejecutar.

La fábrica de software tiene la responsabilidad de realizar encuestas de medición de satisfacción a la Entidad del Estado, sobre los aspectos relativos a la interacción Entidad del Estado -Contratista y sobre la solución funcional y técnica dada a los requerimientos. Asimismo deberá realizar encuestas a su personal sobre la interacción con Entidad del Estado.

La fábrica de software presenta resúmenes sobre la producción y resultados de encuestas de este proceso de trabajo, en Comité de Sistemas, Operativo y Gerencial.

5.1.2.4 ASEGURAMIENTO Y CONTROL DE LA CALIDAD

El Aseguramiento y Control de la Calidad de los Sistemas de Información de la Entidad del Estado, consiste en garantizar la calidad de los productos y de los procesos de Desarrollo y Mantenimiento de Sistemas brindados por el presente Servicio.

La fábrica de software presenta en su metodología el modelo de aseguramiento y control de calidad soportado con las áreas de conocimientos del CMMI indicando por lo menos los objetivos generales y

específicos de la metodología, así como los factores, criterios principales y tipo de acciones preventivas que tomará en el servicio para minimizar la ocurrencia de situaciones como las siguientes:

- Definiciones incompletas, incoherentes y/o ambiguas en los requerimientos por parte del cliente [Definiciones funcionales inadecuadas].
- Especificaciones incompletas, incoherentes y/o ambiguas por parte de la fábrica de software [mala especificación técnica y por ende mala codificación].
- Ausencia de la aplicación sistemática de métodos, procedimientos y normas de ingeniería del software, de parte de algunos o todos los miembros del equipo de desarrollo.
- Escasez, ausencia o uso indebido de entornos integrados de programación.
- Escasez de uso de técnicas actuales y automatizadas para la gestión de proyectos.

Escasez de personal con formación y experiencia en los nuevos métodos, normas y uso de entornos y utilidades de programación.

5.2 DESCRIPCIÓN DEL SISTEMA INVOLUCRADO

El sistema que se tomó para ejecutar la nueva metodología propuesta es el Sistema Nacional de Pensiones el cual realiza el mantenimiento de las solicitudes referidas a pensionamiento y segundos trámites. Dicho mantenimiento abarca desde el ingreso, la calificación, la aprobación, generación de cuentas de pensión y emisión de resoluciones. Administra el pago de pensionistas a través de cuentas de pensión. Esto como parte de los procesos de la subdirección de pago de prestaciones [37] y de la subdirección de calificaciones. En la tabla 5.1, se muestra las características

[37] Una prestación está relacionada a los tipos de pensión que un asegurado tiene derecho. Cabe indicar que el asegurado es la persona que durante su periodo de actividad laboral realiza aportes al Sistema Nacional de Pensiones a fin de gozar de una pensión.

técnicas de dicho sistema; como se puede apreciar, su arquitectura está orientada al modelo Cliente Servidor.

NUEVO SISTEMA DE PENSIONES (NSP)	
LENGUAJE UTILIZADO	Power Builder versión 8.0 Lenguaje C para sistema operativo Unix
BASE DE DATOS	Sistema de Base de Datos • Oracle 9i Enterprise Edition Release 9.2.0.8.0 – 64 bit Sistema Operativo • HP-UX 11i v2 (11.23)
SERVIDOR DE APLICACIONES	Cliente/Servidor Windows Server 2003 R2 Enterprise
SISTEMA OPERATIVO DE RED	Windows Server 2003 R2 Enterprise
SISTEMA OPERATIVO CLIENTE	Windows 98 o superior Adicionalmente se requiere: • Oracle 9i Cliente
INTERFASES	• **Nuevo Sistema de Trámite Documentario (NSTD).** El NSP genera y actualiza la información referente a la calificación como: expedientes, solicitudes, resoluciones, notificaciones. El NSTD consulta esta información para generar reportes y la creación de nuevos expedientes y solitudes. • **Sistema de Recaudación (SISREC).** El NSP proporciona al SISREC los aportes provenientes de las AFP al Sistema Privado de Pensiones como parte del proceso de calificación Libre Desafiliación Informada (LDI). • **Nuevo Sistema de Bonos de Reconocimiento.** El NSBR proporciona al NSP los aportes al Sistema Nacional de Pensiones que son utilizados en los trámites referidos a Bonos de Reconocimiento. • **ONP Virtual.** El NSP proporciona consultas que determinan si una persona es pensionista o no, así como el detalle de los últimos pagos y consultas a determinadas órdenes de pago.

Tabla 5.1: Características Técnicas del Sistema Nacional del Pensiones. [Entidad Pública, 2010]

5.3 ATENCIÓN DE UN REQUERIMIENTO CON LA NUEVA METODOLOGÍA

En la presente tesis se va aplicar la metodología propuesta en la atención de un requerimiento de mantenimiento del Sistema Nacional de Pensiones que a continuación se va a describir.

5.3.1 DESCRIPCIÓN DEL REQUERIMIENTO A DESARROLLAR

Se debe modificar la funcionalidad de depuración de fallecidos[38] y anulación de pagos [masivas e individuales] para que considere también las planillas adicionales[39]; asimismo, en la anulación de pagos debe ingresarse la emisión de pago pudiendo ser la vigente o la última cerrada.

En el Nuevo Sistema de Pensiones [NSP] existe la funcionalidad para anulación de órdenes de pago, estas se realizan por dos opciones:

- Depuración de Fallecidos.
- Anulación de Órdenes de Pago [puntual y masivo].

La anulación de órdenes de pago tanto por depuración de fallecidos y por anulación masiva no considera la anulación de órdenes de pago generadas producto de la ejecución de planillas adicionales, solo anula órdenes de pago de planilla normal para Pensionistas y Alimentistas[40].

A continuación se muestra cómo trabaja actualmente la anulación de órdenes de pago por las opciones antes indicadas:

---

[38] La depuración de fallecidos es un proceso que tiene como objetivo anular las órdenes de pago de aquellos pensionista que durante el proceso de ejecución de las planillas de pago se reportan como fallecidos.

[39] Planilla Adicional es una planilla que se realiza para un caso excepcional con el propósito de pagar a un pensionista que no se le procesó en la planilla normal o para modificar el cálculo del monto a pagar.

[40] Alimentista es aquella persona que goza de una pensión que procede del descuento que se realiza al pensionista por disposición del juez durante el juicio por alimentos. Los hijos no firmados por el padre y nacidos de padres no casados tienen derecho a una pensión alimenticia siempre y cuando, se acredite que la madre tuvo relaciones sexuales con el padre durante la época de la concepción [nueve meses antes del nacimiento].

5.3.1.1 DEPURACIÓN DE PENSIONISTAS FALLECIDOS

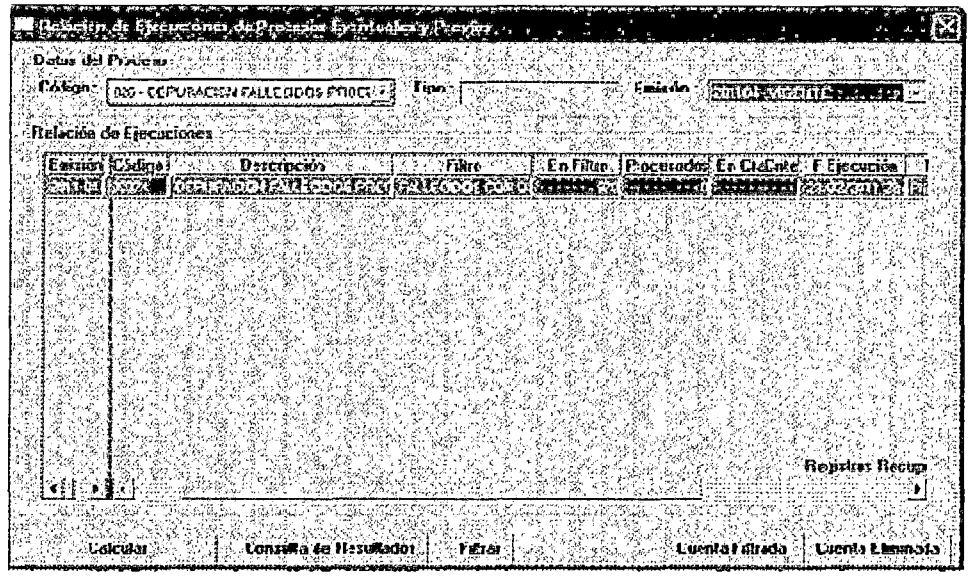


Figura 5.2: Ventana que ejecuta el proceso de Depuración de Fallecidos

El proceso de anulación solo se realiza sobre órdenes de pago correspondientes a planillas normales:

- Para órdenes de pago correspondientes a los regímenes 19990[41] y 18846[42] considera los procesos Pago Pensionistas ó Pago Alimentistas.
- Para órdenes de pago correspondientes al régimen 20530/PPE[43] considera los procesos de planilla normal correspondiente a cada entidad de pago.

5.3.1.1.1 EJECUCIÓN DE FILTRO PARA DEPURACIÓN DE PENSIONISTAS FALLECIDOS

---

[41] El Sistema Nacional de Pensiones fue creado por el Decreto Ley N° 19990 y rige a partir del 1° de mayo de 1973.Se trata de un régimen abierto por cuanto pueden acceder a él los trabajadores provenientes del régimen laboral público y privado, así como los independientes que se afilien en calidad de facultativos. Su administración centralizada se encuentra a cargo de la Oficina Nacional de Normalización Previsional [ONP], a la cual le compete también la administración de otros regímenes pensionarios administrados por el Estado.

[42] Regula el Seguro de Accidentes de Trabajo y Enfermedades Profesionales.

[43] Regula el Régimen de pensiones y compensaciones por servicios civiles prestados al Estado no comprendidos en el Decreto Ley N° 19990. Se paga a pensionistas de entidades disueltas, privatizadas o liquidadas y se califica el derecho en el caso de entidades que pagan pensiones con recursos provenientes del Tesoro Público.

Los criterios de selección registradas para el filtro [44] de 0055-FALLECIDOS CON ORDEN DE PAGO no son empleados por esta.

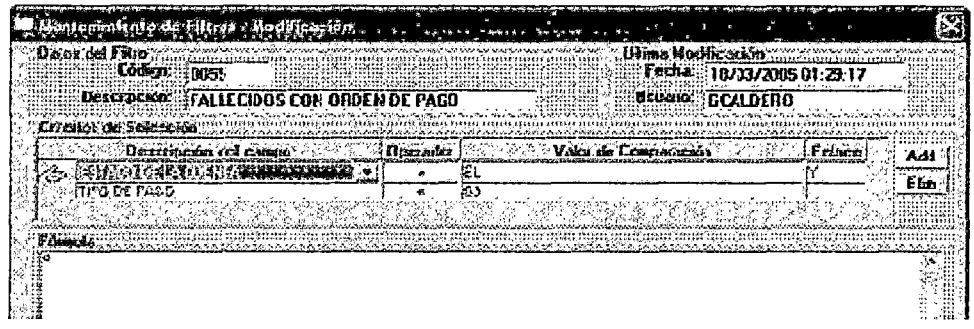


Figura 5.3: Ventana de registro de filtros

La aplicación de criterios se encuentra en el código del programa de ejecución de filtro, dichos criterios son los siguientes:

- Se obtienen las cuentas de pensión que se encuentren en estado eliminadas y motivo de estado:

MOTIVO DE ESTADO
EF- ELIMINADO POR FALLECIMIENTO
FD- ELIMINADO FALLECIDO DEPURADO
FM- FALLECIDO PROCESO MASIVO

- A estas cuentas se consulta si tienen alguna orden de pago de planilla regular o de alimentista, sin considerar el estado ni el tipo de pago[45] en que se encuentren las órdenes de pago, para la emisión[46] de ejecución del proceso Depuración Fallecidos.
- Las cuentas que cumplan con estas condiciones son las que se considerarán para la anulación de sus respectivas órdenes de pago.

[44] Proceso que identifica las cuentas de pensión que fueron eliminadas por fallecimiento que cumplen con los requisitos para anular las órdenes de pago.

[45] El tipo de pago se refiere si los pensionistas cobran por la ventanilla del banco o por cuenta de ahorro o a domicilio.

[46] La emisión esta conformada por el año y mes en la que corresponde el pago de la pensión.

Entonces si se tiene una cuenta en estado fallecido y que solo registra orden de pago por planilla adicional para la emisión de anulación, esta cuenta no será considerada por el filtro.

5.3.1.1.2 EJECUCIÓN DE DEPURACIÓN DE PENSIONISTAS FALLECIDOS

Se anulan las órdenes de pago existentes para la emisión de la ejecución del proceso de depuración de fallecidos. Se actualizan los campos: Estado de Pago, Estado Físico de Pago y Estado de Reintegro al estado Anulado.

En caso la orden de pago corresponda a un Pensionista Fallecido se anula la orden de pago del pensionista y, en caso de existir, se anulan las órdenes de pago de sus alimentistas así como también se anula la retención por consignatarios[47]. Las cuentas de Alimentistas asociadas al pensionista se cambian a estado PA-Paralizado y motivo de estado FP-Fallecimiento Pensionista.

En caso la orden de pago corresponda a un Alimentista fallecido se procede a la anulación de la orden de pago respectiva.

El procedimiento actual de depuración de fallecidos considera, como parte de la anulación de órdenes de pago, revertir el saldo actual y anular la retenciones por consignación existentes.

5.3.1.1.3 MOVIMIENTOS FRACCIONADOS Y SALDO DE DEVENGADOS

En caso de tratarse de movimientos asociados a conceptos de tipo fraccionado el procedimiento de anulación anula la cuota fraccionada pagada en la emisión en que se ejecuta la anulación.

Revierte el saldo del devengado, para ello anula el cargo y realiza el abono generándose el nuevo saldo.

Adicionalmente a la reversión, en caso de tratarse de la última cuota de devengado el procedimiento actualiza el movimiento fraccionado de PA-Pagado a AC-Activo.

---

[47] Persona que recibe en depósito, por auto judicial, el dinero que otro consigna.

5.3.1.1.4 ANULACIÓN DE CONSIGNACIONES

Se realiza la anulación de las consignaciones realizada para la emisión y proceso referida a la orden de pago en la que se realizó la retención, las cuales pueden ser del tipo Retención o Devengado.

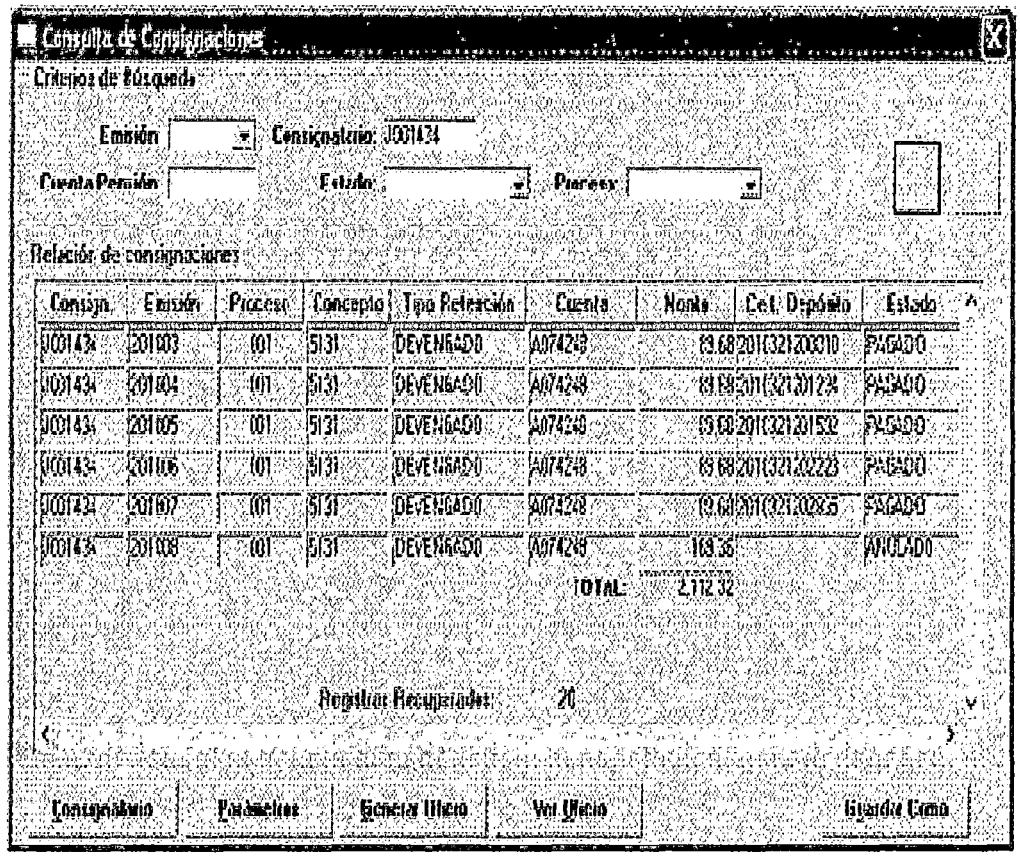


Figura 5. 4: Ventana de consulta de consignaciones generadas por emisión

Las consignaciones anuladas son marcadas para ser mostradas en el Reporte de Depurados por Fallecimiento.

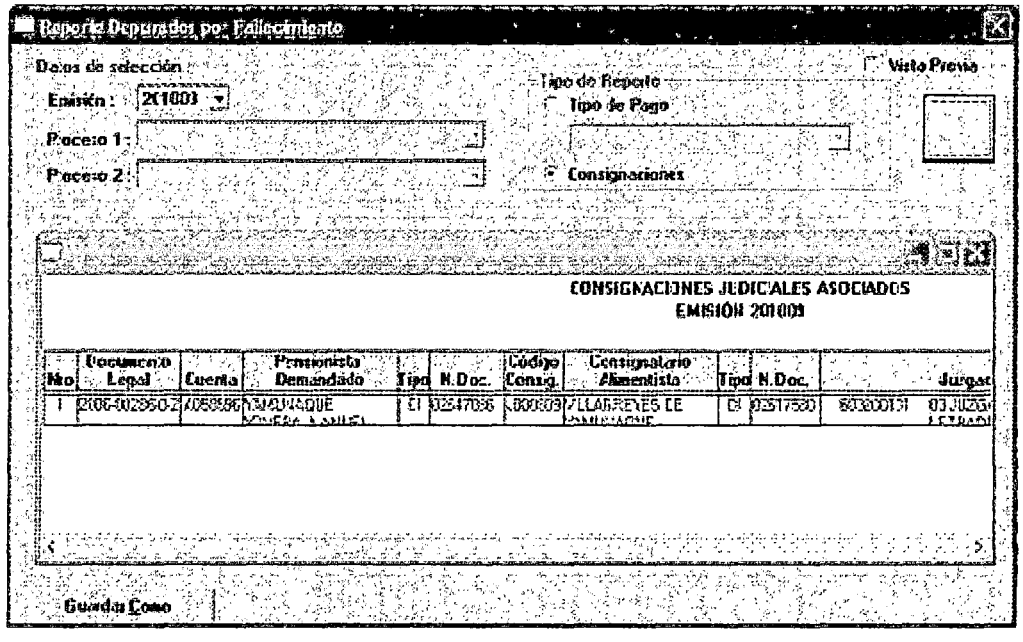


Figura 5.5: Ventana de Reporte de Depurados por Fallecimiento.

En caso de tratarse de una retención tipo Devengado [48], el procedimiento revierte el saldo del devengado, para ello anula el cargo retenido y realiza el abono generándose el nuevo saldo. El nuevo saldo se registra en la tabla de retención de consignatarios.

[48] Deuda que adquiere la entidad del estado, administradora de las pensiones, a favor del pensionista desde la fecha que tiene derecho de pensión hasta la fecha en la que se le otorga.

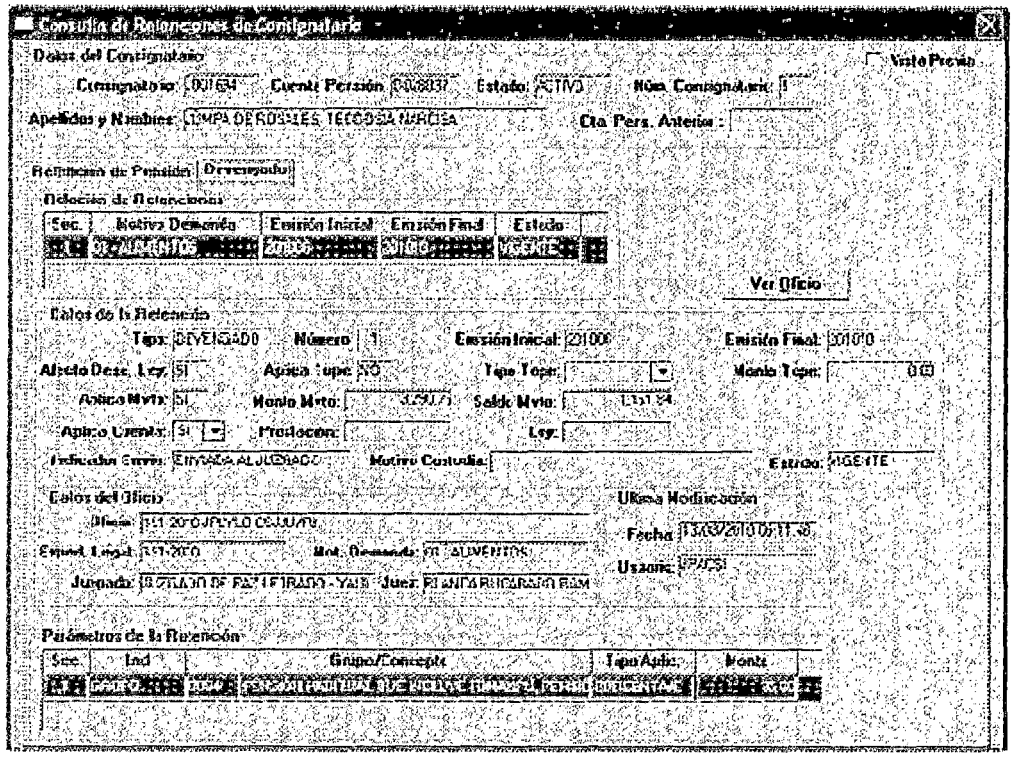


Figura 5.6: Ventana de Consulta de Retenciones y Devengados de Consignatario

El procedimiento anula los movimientos en estado AC-Activo que se hayan registrado y que no se hayan transferido a saldos para la emisión en que se ejecuta la anulación.

5.3.1.2 ANULACIÓN DE ÓRDENES DE PAGO [PUNTUAL Y MASIVO]

La Anulación de órdenes de pago puntual muestra las órdenes de pago registradas cuya emisión coincide con la última emisión en estado CERRADA de la tabla de Emisiones.

La Anulación de orden de pago masiva se realiza a través de la carga de un archivo plano, el cual posee el código de las cuentas a las que se requiere anular sus órdenes de pago, de la misma manera que una anulación puntual se anulan las órdenes de pago existentes para la última emisión en estado CERRADA.

El sistema obtiene el proceso al cual está asociado la orden de pago pero dicho proceso debe corresponder a un proceso de planilla normal [001-

Pago Pensionistas ó 002-Pago Alimentistas] por tanto una orden de pago de planilla adicional no se puede anular.

En caso la orden de pago corresponda a un Pensionista se anula la orden de pago seleccionada y se ejecuta la anulación de la orden de pago de los alimentistas asociados al pensionista.

En caso la orden de pago corresponda a un Alimentista se procede a la anulación de la orden de pago respectiva.

El procedimiento actual de Anulación de Órdenes de Pago considera, como parte de la anulación de órdenes de pago, revertir el saldo actual.

5.3.1.2.1 MOVIMIENTOS FRACCIONADOS Y SALDO DE DEVENGADOS

En caso de tratarse movimientos asociados a conceptos de tipo fraccionado el procedimiento de anulación anula la cuota fraccionada pagada en la emisión en que se ejecuta la anulación.

5.3.1.2.2 ANULACIÓN DE CONSIGNACIONES

El procedimiento no realiza la anulación de consignaciones.Tanto para depuración de fallecidos como para anulación [puntual o masiva] de órdenes de pagos en caso de existir pago de devengados, los procedimientos de anulación revierten el pago de devengado realizado en la emisión sin considerar con qué proceso se realizó el pago de la cuota del devengado, esto se debe a que la lógica de anulación no considera el proceso de pago y además la tabla donde se registra la historia fraccionada no tiene información del proceso lo cual no permite identificar

Para una cuenta que tenga en una misma emisión una planilla regular y una adicional-pago adicional y que en ambos procesos se pague cuota de devengado el proceso de anulación de órdenes de pago solo anulará la orden de pago de planilla regular y anulará la última cuota de devengado pagada la cual corresponde a la cuota de la planilla adicional.

Esto se debe a que el procedimiento lee el último registro tanto de saldos como de historia fraccionada lo cual no es exactamente correcto para la implementación de anulación de órdenes de pago de planillas adicionales.

5.3.1.2.3 REGISTRO DE CÓDIGO DE PROCESO Y EJECUCIÓN EN MOVIMIENTO DE SALDOS

Los campos proceso y ejecución no existen en la tabla de detalle fraccionado de manera que no se puede identificar, para el caso de movimientos fraccionados, qué registro del detalle fraccionado corresponde a la cuota que fue pagada con la orden de pago.

En la tabla de movimiento de saldos se registra el proceso y la ejecución de la planilla que se ejecuta, de manera que se puede identificar en que proceso y ejecución se pagó la cuota de saldo. En caso de tratarse de los siguientes motivos de solicitud de calificaciones el registro de proceso y ejecución en movimiento de saldos se realiza de la siguiente manera:

MOTIVO SOLICITUD	DESCRIPCIÓN
063-Cambio de Tutor	Los procedimientos asociados a este motivo realiza lo siguiente: Copian la historia de saldos y la historia fraccionada del tutor antiguo al tutor nuevo. Se anula el saldo actual y de igual manera la historia fraccionada para el tutor antiguo.
014-Cambio de Riesgo de Invalidez a Jubilación	Los procedimientos asociados a estos motivos copian la historia de saldos y la historia fraccionada de la ley-prestación antigua a la ley-prestación nueva. Se anula el saldo actual y de igual manera la historia fraccionada para la ley-prestación antigua.
015-Cambio de Riesgo	
016-Cambio de Riesgo por Error Material/Boleta de Pago	
121-Modificación o Anulación De Saldos o Movimientos	Los procedimientos asociados a estos motivos realizan lo siguiente: Copian historia de saldos y la historia fraccionada en caso se realice cambio de ley o se cambie el concepto de pago por sentencia judicial. Modifican el saldo, para ello se anula el saldo actual y de igual manera la historia fraccionada.
240-Recalificación Pensión de Jubilación	
250-Regularización por Aplicación de la *Ley 23908 DP*	

Tabla 5.2: Motivos de solicitudes que afectan el proceso de Depuración de Fallecidos

Ejemplo: Se muestra el resultado sobre el movimiento de saldos por aprobación de una solicitud con motivo 016-Cambio de Riesgo que cambia la ley a una cuenta de 01-19990 a 20-25967.

Se copia los movimientos de saldos de la prestación-ley antigua para la nueva prestación-ley.

Se anula el saldo de la prestación-ley antigua [como se indica en color rojo]. Obsérvese que la copia de movimientos de una prestación-ley a otra se mantiene el proceso y ejecución original, y el registro de anulación de saldo para la prestación-ley antigua registra el proceso y ejecución de la última emisión de pago.

CUENTA	PRESTACIÓN	LEY	CONCEPTO	CARGO	ABONO	SALDO	TIPO MOVIMIENTO	PROCESO	EJECUCIÓN	EMISIÓN PROCESO
C469060	02	01	0878	1300.33	0.00	3901.01	04	001	5299	201009
C469060	02	01	0878	0.00	1300.33	5201.34	03			201009
C469060	02	01	0878	1300.33	0.00	3901.01	03	200	0070	201010
C469060	02	01	0878	1300.33	0.00	2600.68	03	205	0019	201010
C469060	02	01	0878	2600.68	0.00	0.00	04	205	0019	201010
C469060	02	20	0878	1300.33	0.00	3901.01	04	001	5299	201009
C469060	02	20	0878	0.00	1300.33	5201.34	03			201009
C469060	02	20	0878	1300.33	0.00	3901.01	03	200	0070	201010
C469060	02	20	0878	1300.33	0.00	2600.68	03	205	0019	201010

Tabla 5.3: Tabla Movimiento Cuenta Corriente Saldos.

En caso de Planilla Adicional-Calculo nuevo se realiza la reversión del saldo, para ello se anula la cuota pagada, se realiza el abono del monto de la cuota anulada y se realiza el pago por la planilla adicional. El registro creado por la reversión de saldo no registra proceso ni ejecución.

Ejemplo: Se muestra el resultado de una planilla normal [001] y una planilla adicional Pago Adicional [205] para la emisión 201010

Obsérvese en la emisión 201008 se realiza el abono por planilla adicional-calculo nuevo, de igual manera en la emisión 201009 se anula la

orden de pago y se abona el nuevo saldo sin proceso ni ejecución. En ambos casos no se registra proceso ni ejecución para los abonos.

CUENTA	SECUENCIA	PRESTACIÓN	LEY	CONCEPTO	CARGO	ABONO	SALDO	TIPO MOVIMIENTO	PROCESO	EJECUCIÓN	EMISIÓN PROCESO
C469060	0003	02	01	0878	1300.33	0.00	5201.34	**04**	001	05250	201008
C469060	0004	02	01	0878	0.00	1300.33	6501.67	03			201008
C469060	0005	02	01	0878	1300.33	0.00	5201.34	03	200	00068	201008
C469060	0006	02	01	0878	1300.33	0.00	3901.01	**04**	001	05299	201009
C469060	0007	02	01	0878	0.00	1300.33	5201.34	03			201009
C469060	0008	02	01	0878	1300.33	0.00	3901.01	03	200	00070	201010
C469060	0011	02	01	0878	1300.33	0.00	2600.68	03	205	00019	201010

Tabla 5.4: Tabla Movimiento Cuenta Corriente Saldos

5.3.2 FASE DE CONCEPCIÓN

5.3.2.1 LISTA DE ITERACIONES

Con el objetivo de descubrir las necesidades que debía satisfacer el sistema que iba a ser desarrollado se destinaron unos días a esta primera fase. En esta etapa, utilizando la ayuda y documentación base aportada por el cliente, se realizó una primera definición sencilla y clara de las características que debía cumplir el requerimiento.

El resultado de esta etapa fue los requerimientos priorizados, artefacto que contiene las historias de usuario a alto nivel que guiarán el proceso de desarrollo y que posteriormente, a medida que avancen las iteraciones, serán descompuestas en un nivel mayor de detalle.

La lista de requerimiento priorizados resultado de esta fase, en síntesis, responde al contenido de la tabla 5.5.

Número	Nombre	Importancia	Estimación Inicial	Cómo probarlo
1	Establecer una relación de dependencia entre la historia fraccionada en movimiento de saldos	1	5	Ejecutar scripts en la base de datos.
2	Modificar el proceso de Depuración de Fallecidos	2	42	Ejecutar filtro para una emisión. Procesar el proceso de depuración para planillas adicionales. Considerar cuentas con historia fraccionada y ejecutar el proceso de depuración de fallecidos. Considerar cuentas con consignatarios y ejecutar el proceso de depuración de fallecidos.
3	Modificar el proceso Anulación de Órdenes de Pago [Puntual y Masivo].	3	15	Considerar cuentas con historia fraccionada y ejecutar el proceso de depuración de fallecidos. Considerar cuentas con consignatarios y ejecutar el proceso de depuración de fallecidos.

Tabla 5.5: Síntesis de los Requerimientos Priorizados resultado de la fase de Concepción.

5.3.2.2 HISTORIAS DEL USUARIO

Número	Historia de Usuario	Descripción
1	Adicionar campos Proceso y Ejecución a tabla Detalle Fraccionado.	• Adicionar los campos proceso y ejecución a la tabla de detalle fraccionado de manera que se establezca correspondencia entre el pago de la cuota en movimiento de saldos y el pago de la cuota en detalle fraccionado. • Permitir la carga de los nuevos campos de la tabla detalle fraccionado de acuerdo a lo siguiente: o Registrar el proceso y ejecución correspondiente a la orden de pago en caso esta realice pago por el concepto del devengado. o Se tomará como referencia las consideraciones para actualizar los campos proceso y ejecución en la

Número	Historia de Usuario	Descripción
		tabla de movimientos de saldos.
2	Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.	• Incluir el registro de proceso y ejecución cuando por motivo de solicitud de calificaciones se realice copia o anulación de historia fraccionada. Los motivos a considerar son: o Cambio de tutor. o Cambio de riesgo. o Modificación o anulación de saldos. o Regularización de pensiones. o Unificación de cuentas.
3	Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo	• Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor. • Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.
4	Modificar Proceso de Depuración de Fallecidos.	• Permitir la anulación de órdenes de pago de planillas adicionales para la opción de depuración de fallecidos. • Modificar el filtro empleado para el proceso 026- Depuración de Fallecidos para que se considere cuentas de pensión que tengan órdenes de pago (normales o adicionales) para la emisión que se ejecutará la depuración. • Las órdenes de pago se anularán de acuerdo a la fecha de ejecución de la carga de la orden de pago en que estas hayan sido generadas. Se anulará empezando por la orden de pago más reciente a la más antigua. • Revertir el saldo de la cuota pagada para la emisión, proceso y ejecución correspondiente. • Dado que se adicionan los campos procesos y ejecución al detalle fraccionado el procedimiento anulará las cuotas correspondientes a la cuenta, prestación, ley, concepto, emisión, proceso y ejecución del detalle fraccionado. • Para el caso de que exista una anulación o modificación de saldos, posterior a la orden de pago a anular, en esta primera etapa, solo se

Número	Historia de Usuario	Descripción
		procederá en caso, dicha modificación o anulación de saldos haya sido originada por una solicitud 121 (modificación o anulación de saldos), en cualquier otro caso no se anulará la orden de pago registrándola en el log de errores. • Modificar el proceso de anulación de consignaciones para que adicionalmente considere el proceso de pago para la anulación de las consignaciones existentes.
5	Modificar Proceso de Anulación de Órdenes de Pago.	• Permitir la anulación de órdenes de pago de planillas adicionales para la opción de anulación de órdenes de pago (puntual y masivo). • En la opción de anulación de órdenes de pago puntual debe mostrar las órdenes de pago de la cuenta de la emisión máxima cerrada o vigente (estado de pagos). • Las órdenes de pago se anularán de acuerdo al orden en que estas hayan sido registradas considerando para ello la fecha de carga cuenta corriente. • No debe permitir anular una orden de pago si existen órdenes de pago posteriores para el pensionista, mostrando un mensaje restrictivo que indique que primero debe anular la orden de pago mas reciente. Esta validación se aplicará a la Anulación de orden de pago Puntual y Masiva. • Considerar el proceso como parte del archivo para anulación de órdenes de pago masivo. • Se obtendrá las órdenes de pago de la emisión máxima cerrada o vigente (estado de pagos) para la cuenta y proceso leída. • Las validaciones sobre las órdenes de pago serán las mismas de la anulación puntual, indicadas en el párrafo anterior. • El proceso de anulación de órdenes de pago por depuración de fallecidos así como anulación puntual y masiva, revertirá el saldo de la cuota pagada en la emisión y proceso correspondiente. • Dado que se adicionan los campos procesos y ejecución al detalle

Número	Historia de Usuario	Descripción
		fraccionado el procedimiento anulará las cuotas correspondientes a la emisión, proceso y ejecución del detalle fraccionado. • Para el caso de que exista una anulación o modificación de saldos, posterior a la orden de pago a anular, en esta primera etapa, solo se procederá en caso, dicha modificación o anulación de saldos haya sido originada por una solicitud 121 (modificación o anulación de saldos), en cualquier otro caso no se anulará la orden de pago registrándola en el log de errores. • Modificar el proceso de anulación de órdenes de pago puntual y masivo para incluir la anulación de las consignaciones existentes.

Tabla 5.6: Historia de Usuario del Requerimiento de Mantenimiento.

5.3.2.3 CRONOGRAMA

Id.	Nombre de tarea	Duración	Comienzo	Fin
1	**Iteración 1: Adicionar campos Proceso y Ejecución a tabla Detalle Fraccionado.**	**5 días**	**02-Feb**	**08-Feb**
2	• Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado.	2 días	02-Feb	03-Feb
3	• Registro del proceso y ejecución por el proceso de carga de cuenta corriente.	3 días	06-Feb	08-Feb
4	**Iteración 2: Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.**	**20 días**	**09-Feb**	**07-Mar**
5	• Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de tutor.	2.5 días	09-Feb	13-Feb
6	• Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de riesgo.	5 días	13-Feb	20-Feb
7	• Registro del proceso y ejecución mediante la aprobación del motivo de solicitud modificación y anulación de saldos.	5 días	20-Feb	27-Feb
8	• Registro del proceso y ejecución mediante la aprobación del motivo de	5 días	27-Feb	05-Mar

Id.	Nombre de tarea	Duración	Comienzo	Fin
	regularización de pensiones.			
9	• Registro del proceso y ejecución mediante la aprobación del motivo de unificación de cuentas.	2.5 días	05-Mar	07-Mar
10	**Iteración 3: Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo.**	**5 días**	**08-Mar**	**14-Mar**
11	• Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor.	3 días	08-Mar	12-Mar
12	• Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.	2 días	13-Mar	14-Mar
13	**Iteración 4: Modificar Proceso de Depuración de Fallecidos.**	**10 días**	**15-Mar**	**28-Mar**
14	• Modificación del Filtro.	2 días	15-Mar	16-Mar
15	• Anulación de la Orden de Pago.	1 día	19-Mar	19-Mar
16	• Revertir Saldo y Cuota de Devengado Planilla Normal incluye la ley 27803.	2 días	20-Mar	21-Mar
17	• Revertir Saldo y Cuota de Devengado Planilla Adicionales.	1 día	22-Mar	22-Mar
18	• Anulación de Consignaciones por medio de la opción de depuración de fallecidos y anulación de órdenes de pago puntual y masivo.	2 días	23-Mar	26-Mar
19	• Anulación de consignaciones mediante la opción del módulo de consignaciones.	2 días	27-Mar	28-Mar
20	**Iteración 5: Modificar Proceso de Anulación de Órdenes de Pago.**	**15 días**	**29-Mar**	**18-Abr**
21	• Anulación de Órdenes de Pago Puntual.	5 días	29-Mar	04-Abr
22	• Anulación de órdenes de Pago Masiva.	2 días	05-Abr	06-Abr
23	• Revertir saldo y cuota de devengado.	3 días	09-Abr	11-Abr
24	• Anulación de Consignaciones.	5 días	12-Abr	18-Abr

5.3.2.4 PLAN DE ALTO NIVEL

Número	Historia de Usuario	Esfuerzo [días]
1	Adicionar campos Proceso y Ejecución a tabla Detalle Fraccionado.	5
2	Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.	20
3	Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo.	5
4	Modificar Proceso de Depuración de Fallecidos.	10
5	Modificar Proceso de Anulación de Órdenes de Pago.	15

5.3.3 FASE DE ELABORACIÓN

5.3.3.1 HISTORIA DE USUARIO ESPECIFICADO

Número	Usuario	Nombre de la Historia	Prioridad en negocio	Riesgo en Desarrollo	Puntos estimados	Iteración asignada	Programador asignado	Descripción
1	Supervisor de Planillas	Adicionar campos Proceso y Ejecución a tabla Detalle Fraccionado.	Alta	Bajo	5	1	Programador 1	1. Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado. 2. Registro del proceso y ejecución por el proceso de carga de cuenta corriente.

Número	Usuario	Nombre de la Historia	Prioridad en negocio	Riesgo en Desarrollo	Puntos estimados	Iteración asignada	Programador asignado	Descripción
2	Aprobador de Solicitudes	Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.	Alta	Alta	20	2	Programador 2	3. Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de tutor. 4. Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de riesgo. 5. Registro del proceso y ejecución mediante la aprobación del motivo de solicitud modificación y anulación de saldos. 6. Registro del proceso y ejecución mediante la aprobación del motivo de regularización de pensiones. 7. Registro del proceso y ejecución mediante la aprobación del motivo de unificación de cuentas.
3	Operador de Centro de Cómputo	Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo.	Alta	Medio	5	3	Programador 1	8. Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor. 9. Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.

Número	Usuario	Nombre de la Historia	Prioridad en negocio	Riesgo en Desarrollo	Puntos estimados	Iteración asignada	Programador asignado	Descripción
4	Operador de Centro de Cómputo	Modificar Proceso de Depuración de Fallecidos.	Alta	Alto	10	4	Programador 2	10. Modificación del Filtro. 11. Anulación de la Orden de Pago. 12. Revertir Saldo y Cuota de Devengado Planilla Normal incluye la ley 27803. 13. Revertir Saldo y Cuota de Devengado Planilla Adicionales. 14. Anulación de Consignaciones por medio de la opción de depuración de fallecidos y anulación de órdenes de pago puntual y masivo. 15. Anulación de consignaciones mediante la opción del módulo de consignaciones.
5	Operador de Centro de Cómputo	Modificar Proceso de Anulación de Órdenes de Pago.	Alta	Alto	15	5	Programador 1	16. Anulación de Órdenes de Pago Puntual. 17. Anulación de órdenes de Pago Masiva. 18. Revertir saldo y cuota de devengado. 19. Anulación de Consignaciones.

5.3.3.2 PLAN DE VERSIÓN

Número	Historia de Usuario	Prioridad	Riesgo	Esfuerzo [días]	Iteración
1	Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado.	Media	Bajo	5	1
2	Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.	Alta	Alta	20	2

Número	Historia de Usuario	Prioridad	Riesgo	Esfuerzo [días]	Iteración
3	Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo.	Alta	Medio	5	3
4	Modificar Proceso de Depuración de Fallecidos.	Alta	Alto	10	4
5	Modificar Proceso de Anulación de Órdenes de Pago.	Alta	Alto	15	5

5.3.3.3 ARQUITECTURA TÉCNICA. DIAGRAMA DE COMPONENTES QUE PARTICIPAN DEL PROCESO DE ANULACIÓN DE ÓRDENES DE PAGO

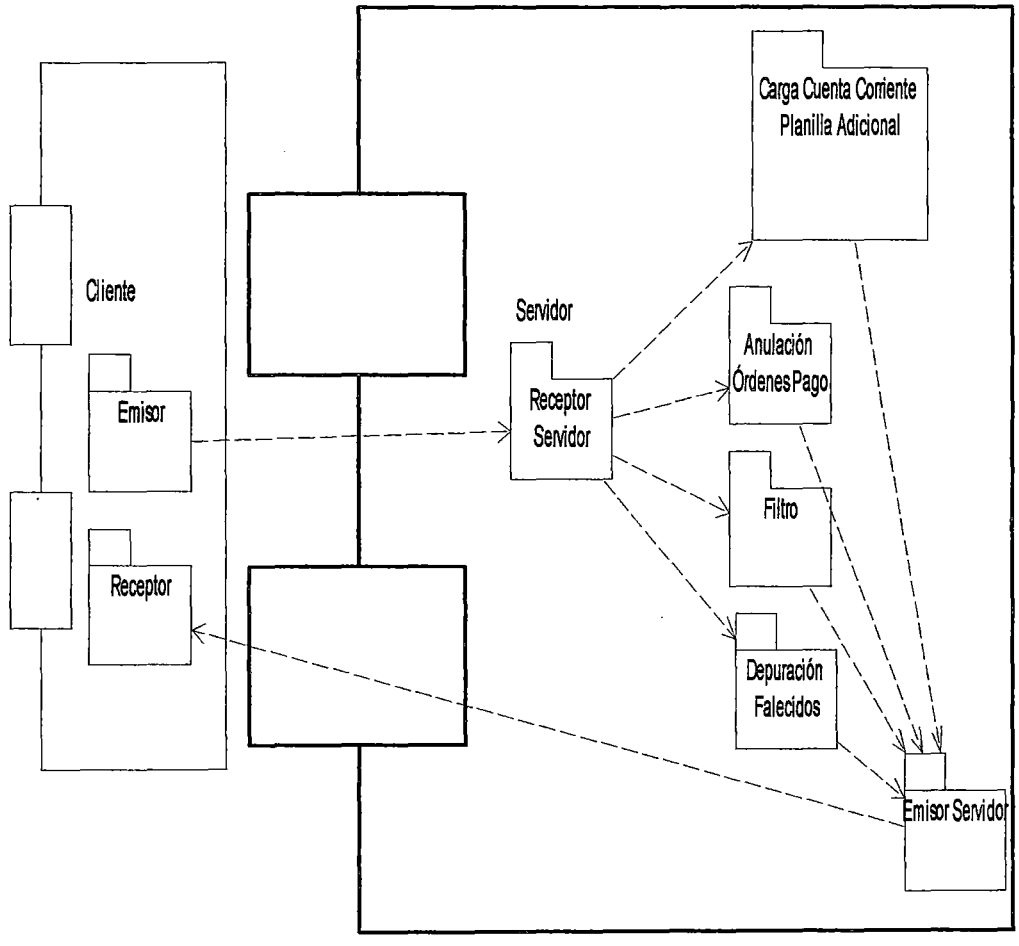


Figura 5. 7: Diagrama de Componentes del Proceso de Anulación de Órdenes de Pago

5.3.3.4 ARQUITECTURA TÉCNICA. DIAGRAMA DE DESPLIEGUE

En el siguiente diagrama se muestra la cobertura a nivel nacional del Nuevo Sistema de Pensiones.

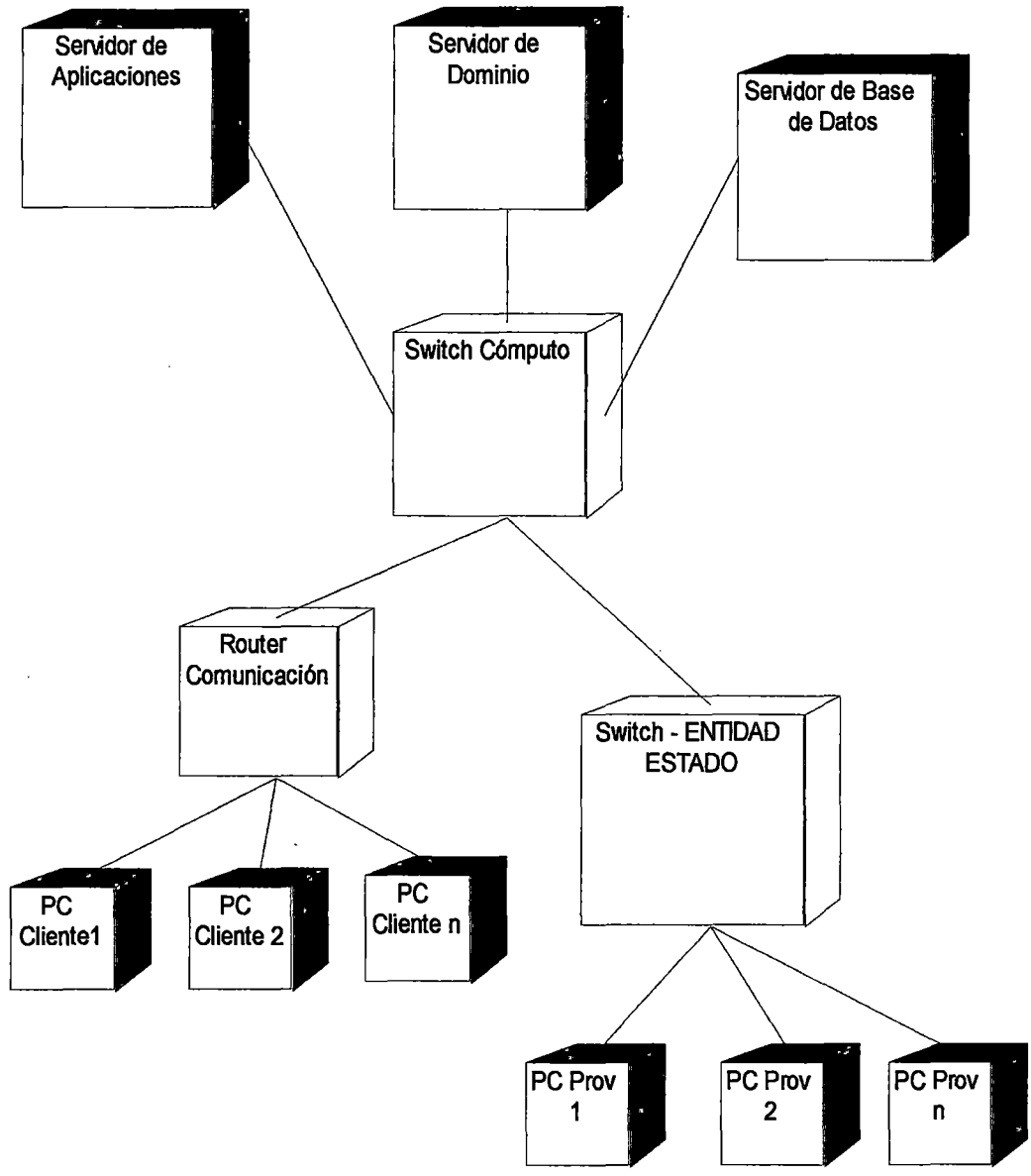


Figura 5. 8: Diagrama de Despliegue del Nuevo Sistema de Pensiones.

5.3.4 FASE DE CONSTRUCCIÓN

5.3.4.1 TAREA DE INGENIERÍA

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
1	1	Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado.	Desarrollo	2	02-febrero	03-febrero	Programador 1	Los campos nuevos se cargarán considerando lo siguiente: • En caso de cálculo de pago tendrán el valor del proceso y ejecución de la emisión de pago. • En caso de anulación de saldos el registro de anulación de historia fraccionada se colocará los valores de proceso y ejecución del último registro del detalle fraccionado.
2	1	Registro del proceso y ejecución por el proceso de carga de cuenta corriente.	Desarrollo	3	06-febrero	08-febrero	Programador 1	Para anulación de orden de pago por depuración de fallecidos o anulación de orden de pago puntual y masiva los registros generados por reversión en movimiento de saldos no registrarán valores para proceso y ejecución.
1	2	Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de tutor.	Desarrollo	2.5	09-febrero	13-febrero	Programador 2	Al copiar la historia de saldos y la historia fraccionada del tutor antiguo al tutor nuevo actualizar el código de proceso y ejecución.
2	2	Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de riesgo.	Desarrollo	5	13-febrero	20-febrero	Programador 2	Actualizar el código de proceso y número de ejecución cuando se copian la historia de saldos y la historia fraccionada de la ley-prestación antigua a la ley-prestación nueva.

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
3	2	Registro del proceso y ejecución mediante la aprobación del motivo de solicitud modificación y anulación de saldos.	Desarrollo	5	20-febrero	27-febrero	Programador 2	Actualizar el código de proceso y número de ejecución al momento que se cambie el concepto por sentencia judicial.
4	2	Registro del proceso y ejecución mediante la aprobación del motivo de regularización de pensiones	Desarrollo	5	27-febrero	05-marzo	Programador 2	Actualizar el código de proceso y número de ejecución cuando se copian historia de saldos y la historia fraccionada en caso se realice cambio de ley o se cambie el concepto de pago por sentencia judicial.
5	2	Registro del proceso y ejecución mediante la aprobación del motivo de unificación de cuentas.	Desarrollo	2.5	05-marzo	07-marzo	Programador 2	Actualizar el código de ejecución y código de proceso al momento que se copia la historia fraccionada a la cuenta receptora.
1	3	Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor.	Desarrollo	3	08-marzo	12-marzo	Programador 1	Obtener el saldo de la tabla temporal de movimiento cuenta corriente saldos y actualizar en el monto de saldo de la historia fraccionada.
2	3	Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.	Desarrollo	2	13-marzo	14-marzo	Programador 1	Leer la tabla movimiento cuenta corriente y validar todos los conceptos que se tienen saldos si existiere un concepto para una determinada prestación y ley tiene su saldo anulado entonces quiere decir que hubo alguna aprobación de un motivo por lo tanto se emite un mensaje para realizar la reversión en forma manual.

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
1	4	Modificación del Filtro.	Desarrollo	2	15-marzo	16-marzo	Programador 2	Se modificará el filtro empleado para el proceso de depuración de fallecidos para que considere cuentas de pensión con orden de pago para la emisión de depuración de fallecidos.
2	4	Anulación de la Orden de Pago.	Desarrollo	1	19-marzo	19-marzo	Programador 2	El cambio a realizar permitirá que se realice la Anulación de órdenes de pago por depuración de fallecidos. La anulación de las órdenes de pago se realizará de acuerdo a la fecha de carga cuenta corriente para el proceso empezando por la de fecha más reciente a la más antigua.
3	4	Revertir Saldo y Cuota de Devengado Planilla Normal incluye la ley 27803.	Desarrollo	2	20-marzo	21-marzo	Programador 2	En caso se realice una modificación o anulación de saldos posterior a la orden de pago que se requiere anular el procedimiento no anulará la orden de pago registrando en el log de errores el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago". Se anulará el registro de la cuota pagada de la tabla de movimiento de saldos para la cuenta-prestación-ley-concepto del proceso y ejecución asociado a la orden de pago. Se realizará el abono de la cuota anulada, adicionado al saldo pendiente, en la tabla de movimiento de saldos para la cuenta-prestación-ley-concepto del proceso y emisión asociado a la orden de pago. El registro generado por abono de la cuota anulada en movimiento de saldos no registrará valores para proceso y ejecución. Se anulará el registro de la cuota pagada de la tabla de detalle fraccionado para la cuenta-prestación-ley-concepto del proceso y emisión asociado a la orden de pago.

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
4	4	Revertir Saldo y Cuota de Devengado Planilla Adicionales.		1	22-marzo	22-marzo	Programador 2	Se obtendrá los registros de la tabla de movimientos de saldos correspondientes a la cuenta-prestación-ley-concepto del proceso y emisión asociado a las órdenes de pago que se anularán (Tabla Resumen Cuenta Corriente). Se anulará cada uno de los registros de saldos obtenidos en el paso anterior. Se adicionará un registro en la tabla de movimiento de saldos por cada registro a anular. En caso de movimientos fraccionados se seleccionará los registros correspondientes a la cuenta-emisión-prestación-ley-concepto proceso y ejecución asociados a la orden de pago a anular. Se anulará los registros seleccionados de historia fraccionada. En caso de tratarse de la última cuota se actualiza el movimiento fraccionado revirtiéndose del estado PA-Pagado a AC-Activo. Para el caso que la el saldo de se haya anulado entonces se emitirá un mensaje para que la reversión se haga manualmente.
5	4	Anulación de Consignaciones por medio de la opción de depuración de fallecidos y anulación de órdenes de pago puntual y masivo.	Desarrollo	2	23-marzo	26-abril	Programador 2	Se modificará el proceso de anulación de consignaciones para incluir el proceso de pago de manera que se guarde correlación entre la de orden de pago anulada y la consignación.

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
6	4	Anulación de consignaciones mediante la opción del módulo de consignaciones		2	27-marzo	28-marzo	Programador 2	Los montos de los movimientos para el concepto de reintegro deberán acumularse si existiera varios procesos asociados a un consignatario para una determinada emisión.
1	5	Anulación de Órdenes de Pago Puntual	Desarrollo	5	29-marzo	04-abril	Programador 1	Mostrar las órdenes de pago de cuenta para la máxima emisión cerrada o vigente considerando la fecha de ejecución más reciente de la tabla ejecución para el proceso y ejecución asociado a la orden de pago a anular. La anulación de órdenes de pago se realizará de acuerdo a la fecha de ejecución de la orden de pago en que estas se hayan registrado validándose que no se pueda anular una planilla regular si existe adicionales vigentes. En caso de existir más de una planilla adicional se validará que se anule primero la adicional mas reciente. Para anulación masiva se incluirá el proceso de pago como parte del archivo de carga. Se aplicará las mismas consideraciones para anulación puntual indicadas en el párrafo.

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
2	5	Anulación de órdenes de Pago Masiva	Desarrollo	2	05-abril	06-abril	Programador 1	Considerar el proceso como parte del archivo para anulación de órdenes de pago masivo. El formato será el siguiente: XXXXXXXYYY, donde: XXXXXXX: Código de cuenta pensión y YYY: Código de proceso de planilla. En caso se requiera anular todas las órdenes de pago para una cuenta enviar como proceso XXX. Se ordenará en orden descendiente las cuentas de acuerdo a la fecha de carga cuenta corriente de las órdenes de pago, este dato se encuentra en la tabla de ejecuciones de pago de NSP. No debe permitir anular una orden de pago si existen órdenes de pago posteriores para el pensionista, se registrará en el log de errores la cuenta no anulada y el mensaje:"No se puede anular planilla seleccionada, seleccione planilla adicional más reciente". Se validará que no exista modificación o anulación de saldos posterior a la orden de pago que se requiere anular para ellos se obtendrá el último movimiento de saldo de la tabla movimiento cuenta corriente saldo, para la cuenta-prestación-ley-concepto en caso el valor del campo fecha de movimiento de dicho registro es diferente al campo fecha de movimiento del movimiento de saldo a anular el procedimiento anulará la orden de pago y registrará la cuenta en el log de errores el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago hacer la reversión de saldos manualmente".

Número de tarea de ingeniería	Número de historia de usuario	Nombre de la tarea	Tipo de tarea	Puntos estimados	Fecha de inicio	Fecha fin	Programador responsable	Descripción
3	5	Revertir saldo y cuota de devengado	Desarrollo	3	09-abril	11-abril	Programador 1	Se obtendrá los registros de la tabla de movimientos de saldos correspondientes a la cuenta-prestación-ley-concepto del proceso y emisión asociado a las órdenes de pago que se anularán (Tabla Resumen Cuenta Corriente). Se anulará cada uno de los registros de saldos obtenidos en el paso anterior. Se adicionará un registro en la tabla de movimiento de saldos por cada registro a anular. En caso de movimientos fraccionados se seleccionará los registros correspondientes a la cuenta-emisión-prestación-ley-concepto proceso y ejecución asociados a la orden de pago a anular. Se anulará los registros seleccionados de historia fraccionada. En caso de tratarse de la última cuota se actualiza el movimiento fraccionado revirtiéndose del estado PA-Pagado a AC-Activo.
4	5	Anulación de Consignaciones	Desarrollo	5	12-abril	18-abril	Programador 1	El proceso de anulación de consignaciones se incluirá en el proceso de anulación de orden de pago puntual y masivo. A diferencia de la anulación por fallecimiento la consignación no se marcará como Intervenida.

Tabla 5. 7: Tareas de Ingeniería por Historia de Usuario del Requerimiento de Mantenimiento.

5.3.4.2 PLAN DE ITERACIÓN

Número de historia de usuario	Número de tarea de ingeniería	Fecha inicio	Fecha fin	Programador
1	1	02-febrero	03-febrero	Programador 1

Número de historia de usuario	Número de tarea de ingeniería	Fecha inicio	Fecha fin	Programador
1	2	06-febrero	08-febrero	Programador 1
2	1	09-febrero	13-febrero	Programador 2
2	2	13-febrero	20-febrero	Programador 2
2	3	20-febrero	27-febrero	Programador 2
2	4	27-febrero	05-marzo	Programador 2
2	5	05-marzo	07-marzo	Programador 2
3	1	08-marzo	12-marzo	Programador 1
3	2	13-marzo	14-marzo	Programador 1
4	1	15-marzo	16-marzo	Programador 2
4	2	19-marzo	19-marzo	Programador 2
4	3	20-marzo	21-marzo	Programador 2
4	4	22-marzo	22-marzo	Programador 2
4	5	23-marzo	26-marzo	Programador 2
4	6	27-marzo	28-marzo	Programador 2
5	1	29-marzo	04-abril	Programador 1
5	2	05-abril	06-abril	Programador 1
5	3	09-abril	11-abril	Programador 1
5	4	12-abril	18-abril	Programador 1

5.3.4.3 CASOS DE PRUEBA DE ACEPTACIÓN

Número de historia de usuario	Requisito	Número de caso de prueba	Caso de prueba
1	Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado.	1	Validar que se haya grabado el proceso y la ejecución en el último registro del detalle de la historia fraccionada luego de anular un saldo.
		2	Comprobar que luego de la anulación de una orden de pago los registros generados por reversión en movimiento cuenta corriente saldo no registrarán valores para proceso y ejecución.
		3	Validar que se actualice los campos código de proceso y número de ejecución luego de culminar la carga de cuenta corriente tanto para la planilla normal como para la planilla adicional.
2	Actualizar los campos de Proceso y Ejecución en la aprobación de los motivos de solicitud.	1	Verificar que luego de la aprobar un motivo de solicitud cambio de tutor copie la historia de saldos y la historia fraccionada del tutor antiguo al tutor nuevo.
		2	Verificar cuando se apruebe el motivo de solicitud cambio de riesgo, que el código de proceso y número de

Número de historia de usuario	Requisito	Número de caso de prueba	Caso de prueba
			ejecución se incluya cuando se copian la historia de saldos y la historia fraccionada de la ley-prestación antigua a la ley-prestación nueva.
		3	Validar que cuando se apruebe el motivo de solicitud de modificación o anulación de saldos, el código de proceso y número de ejecución se registren al momento que se cambie el concepto por sentencia judicial.
		4	Comprobar cuando se apruebe el motivo de solicitud de regularización de pensiones se registren el código de proceso y número de ejecución cuando se copian historia de saldos y la historia fraccionada en caso se realice cambio de ley o se cambie el concepto de pago por sentencia judicial.
		5	Validar que cuando se apruebe el motivo de solicitud de unificación se registren el código de ejecución y código de proceso al momento que se copia la historia fraccionada a la cuenta receptora.
3	Anulación de órdenes de pago por la Planilla Adicional Nuevo Cálculo.	1	Validar para el caso de aprobación de motivo se solicitud, modificación o anulación de saldo, antes de aplicar la planilla adicional, la carga de cuenta corriente actualice el último saldo de la historia de saldos.
		2	Validar que para el saldo pagado por una planilla adicional si esta haya sido anulada entonces se muestre un mensaje que la reversión se tiene que hacer en forma manual.
4	Modificar Proceso de Depuración de Fallecidos.	1	Validar que para una cuenta que solo tenga una planilla adicional para una determinada emisión el proceso de filtro para la depuración de fallecidos lo considere para anular su orden de pago.
		2	Comprobar para el caso de una cuenta de pensión tenga para una emisión planillas tanto la normal como adicional que la anulación lo haga comenzando con la fecha más reciente.
		3	Validar que la orden de pago se anule

Número de historia de usuario	Requisito	Número de caso de prueba	Caso de prueba
			y en el log de errores salga el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago" si se detecta que se realizó una modificación o anulación posterior a la orden de pago.
		4	Verificar que la anulación del registro de saldos sea de la cuenta-prestación-ley-concepto del proceso y ejecución asociado a la orden de pago.
		5	Validar la reversión de la historia de saldos y de la historia fraccionada.
		6	Comprobar que para el caso de consignaciones la anulación sea para el proceso y ejecución correspondiente y que la consignación se marque como intervenida.
5	Modificar Proceso de Anulación de Órdenes de Pago.	1	Validar para una anulación de una orden de pago puntual o masivo que me permita anular la orden de pago más reciente.
		2	Comprobar para el caso de una anulación masiva si en el archivo de carga envío el código de proceso como XXX el proceso anula todas las órdenes de pago asociada a la cuenta en una emisión determinada.
		3	Validar que no se anula la orden de pago si existe alguna posterior.
		4	Validar que se anule la orden de pago y que en el log de errores salga el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago hacer la reversión de saldos manualmente" si se detecta que se realizó una modificación o anulación posterior a la orden de pago.
		5	Verificar que la anulación del registro de saldos sea de la cuenta-prestación-ley-concepto del proceso y ejecución asociado a la orden de pago.
		6	Validar la reversión de la historia de saldos y de la historia fraccionada.
		7	Comprobar que se haya actualizado con el estado AC- Activo si el estado anterior fue PA-Pagado.
		8	Comprobar que para el caso de

Número de historia de usuario	Requisito	Número de caso de prueba	Caso de prueba
			consignaciones la anulación sea para el proceso y ejecución correspondiente y que la consignación se marque como intervenida.

5.3.4.4 TARJETA CRC

Clases:

- Cuenta.pensión
- Pensionista Dependiente
- Historia de Saldos
- Historia Fraccionada
- Detalle de la Historia Fraccionada
- Orden de Pago
- Alimentistas
- Consignatario
- Recuperos

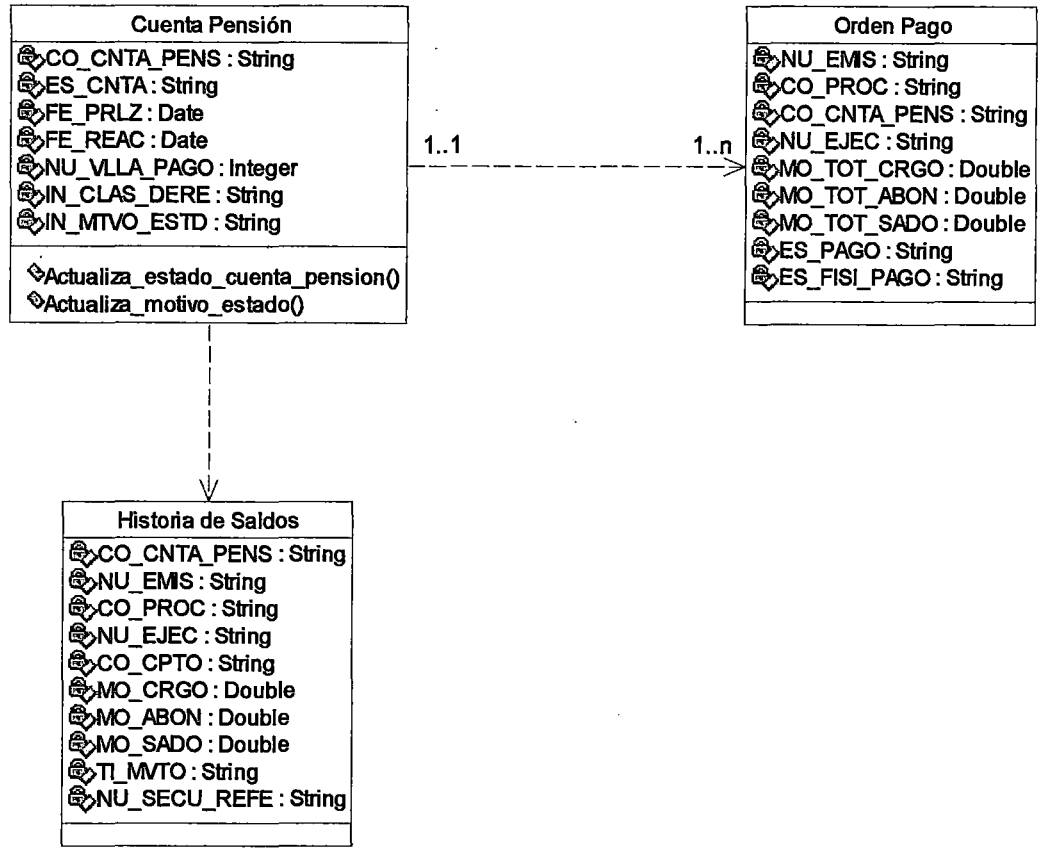


Figura 5. 9: Diagrama de Clases del Proceso de Anulación de Órdenes de Pago.

Código de Tarjeta	Clase	Responsabilidades	Colaboradores
Fallecer pensionista	Cuenta Pensión	• Actualizar el estado de cuenta. • Actualizar el indicador de motivo de estado. • Registrar recuperos.	Pensionista Dependiente Recuperos

Código de Tarjeta	Clase	Responsabilidades	Colaboradores
Depuración de Fallecidos	Orden de Pago	• Filtrar cuentas eliminadas por fallecimiento. • Actualizar el estado de pago a AN – Anulado. • Anular Orden de Pago de Alimentista • Anular Consignatarios • Revertir los saldos • Revertir historia fraccionada • Actualizar el campo última emisión de pago.	Historia de Saldos Historia Fraccionada Detalle de la Historia Fraccionada Cuenta Pensión Alimentista Consignatario

Código de Tarjeta	Clase	Responsabilidades	Colaboradores
Anulación Individual orden de pago	Orden de Pago	• Anular el proceso más reciente. • Actualizar el estado de pago a AN – Anulado. • Anular Orden de Pago de Alimentista. • Anular Consignatarios. • Revertir los saldos. • Revertir historia fraccionada. • Actualizar el campo última emisión de pago de la cuenta de pensión.	Historia de Saldos Historia Fraccionada Detalle de la Historia Fraccionada Cuenta Pensión Alimentista Consignatario

5.3.4.5 CASOS DE USO

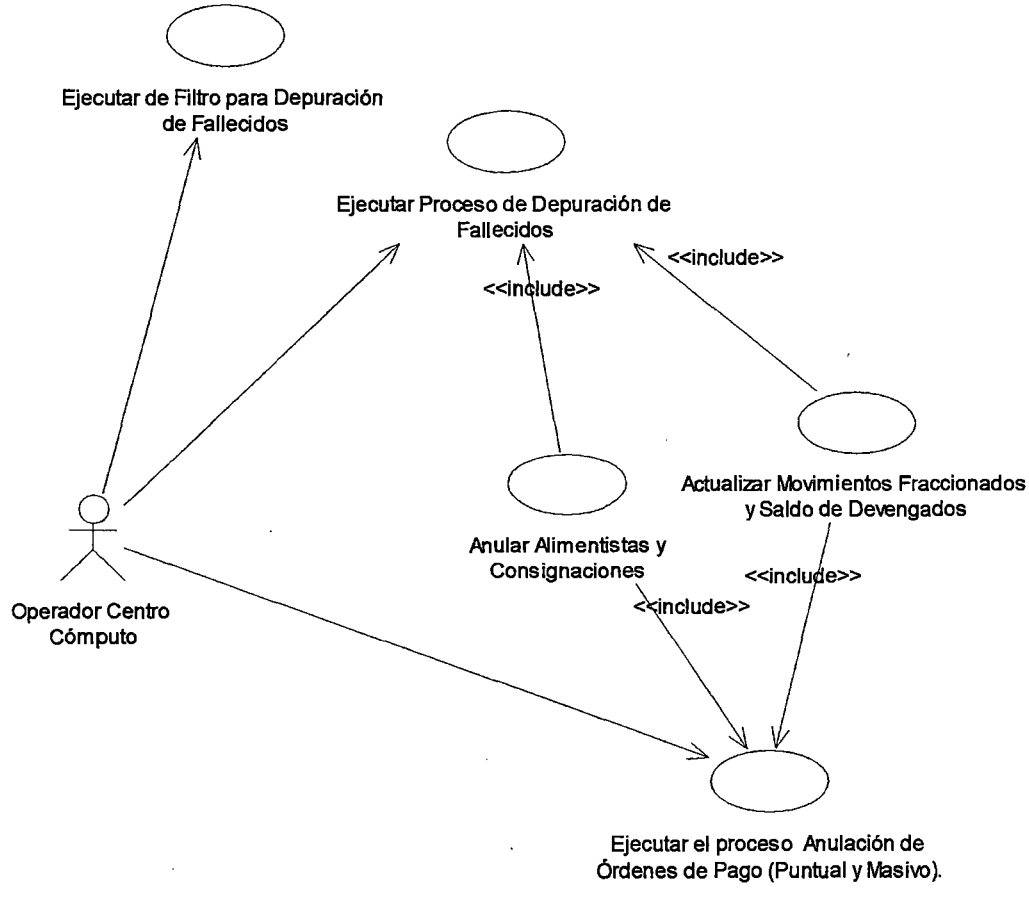


Figura 5.10: Diagrama de Casos de Uso del Proceso de Anulación de Órdenes de Pago

5.3.4.6 REPORTE DE PRUEBAS DE ACEPTACIÓN

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
1	1	Validar que se haya grabado el proceso y la ejecución en el último registro del detalle de la historia fraccionada luego de anular un saldo.	Ejecutar el proceso de anulación de orden de pago (individual, masivo o depuración de fallecidos o carga de cuenta corriente nuevo cálculo).	Emisión, código de proceso, cuenta de pensión.	Se grabe el proceso y ejecución en la anulación del saldo.	Se grabó el proceso y ejecución.
2	1	Comprobar que luego de la anulación de una orden de pago los registros generados por reversión en movimiento cuenta corriente saldo no registrarán valores para proceso y ejecución.	Ejecutar el proceso de anulación de orden de pago (individual, masivo o depuración de fallecidos o carga de cuenta corriente nuevo cálculo).	Emisión, código de proceso, cuenta de pensión.	El registro de abono no cuenta con el grabe el proceso ni ejecución.	No se registró el código de proceso y ejecución en movimiento cuenta corriente saldos.
3	1	Validar que se actualice los campos código de proceso y número de ejecución luego de culminar la carga de cuenta corriente tanto para la planilla normal como para la planilla adicional.	Ejecutar la carga de cuenta corriente.	Código de proceso, número de ejecución y número de emisión.	Se actualice en la tabla movimiento fraccionado los campos código de proceso y número de ejecución.	Se actualizó los campos código de proceso y número de ejecución.
1	2	Verificar que luego de la aprobar un motivo de solicitud cambio de tutor copie la historia de saldos y la historia fraccionada del tutor antiguo al tutor nuevo	Ejecutar la aprobación de un motivo de solicitud.	Motivo de solicitud, número de expediente, código de cuenta y datos adicionales.	Cuando copia la historia fraccionada incluya al código de proceso y número de ejecución.	Los campos proceso y ejecución se actualizaron en la tabla de movimiento fraccionado detalle.
2	2	Verificar cuando se apruebe el motivo de solicitud cambio	Ejecutar la aprobación de un motivo de	Motivo de solicitud, número de	Cuando copia la historia fraccionada	Los campos proceso y ejecución se

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
		de riesgo, que el código de proceso y número de ejecución se incluya cuando se copian la historia de saldos y la historia fraccionada de la ley-prestación antigua a la ley-prestación nueva.	solicitud.	expediente, código de cuenta y datos adicionales.	incluya al código de proceso y número de ejecución.	actualizaron en la tabla de movimiento fraccionado detalle.
3	2	Validar que cuando se apruebe el motivo de solicitud de modificación o anulación de saldos, el código de proceso y número de ejecución se registren al momento que se cambie el concepto por sentencia judicial.	Ejecutar la aprobación de un motivo de solicitud.	Motivo de solicitud, número de expediente, código de cuenta y datos adicionales.	Cuando copia la historia fraccionada incluya al código de proceso y número de ejecución.	Los campos proceso y ejecución se actualizaron en la tabla de movimiento fraccionado detalle.
4	2	Comprobar cuando se apruebe el motivo de solicitud de regularización de pensiones se registren el código de proceso y número de ejecución cuando se copian historia de saldos y la historia fraccionada en caso se realice cambio de ley o se cambie el concepto de pago por sentencia judicial.	Ejecutar la aprobación de un motivo de solicitud.	Motivo de solicitud, número de expediente, código de cuenta y datos adicionales.	Cuando copia la historia fraccionada incluya al código de proceso y número de ejecución.	Los campos proceso y ejecución se actualizaron en la tabla de movimiento fraccionado detalle.
5	2	Validar que cuando se apruebe el motivo de solicitud de unificación se registren el código de ejecución y	Ejecutar la aprobación de un motivo de solicitud.	Motivo de solicitud, número de expediente, código de cuenta y	Cuando copia la historia fraccionada incluya al código de proceso y	Los campos proceso y ejecución se actualizaron en la tabla de movimiento

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
		código de proceso al momento que se copia la historia fraccionada a la cuenta receptora.		datos adicionales.	número de ejecución.	fraccionado detalle.
1	3	Validar para el caso de aprobación de motivo se solicitud, modificación o anulación de saldo, antes de aplicar la planilla adicional, la carga de cuenta corriente actualice el último saldo de la historia de saldos.	Ejecutar la aprobación de un motivo de solicitud.	Motivo de solicitud, número de expediente, código de cuenta y datos adicionales.	Cuando copia la historia fraccionada incluya al código de proceso y número de ejecución.	Los campos proceso y ejecución se actualizaron en la tabla de movimiento fraccionado detalle.
2	3	Validar que para el saldo pagado por una planilla adicional si esta haya sido anulada entonces se muestre un mensaje que la reversión se tiene que hacer en forma manual.	Ejecutar el proceso de anulación de orden de pago (individual, masivo o depuración de fallecidos o carga de cuenta corriente nuevo cálculo).	Emisión, código de proceso, cuenta de pensión.	Se muestre un mensaje que la reversión se tiene que hacer en forma manual.	Se mostró el mensaje esperado.
1	4	Validar que para una cuenta que solo tenga una planilla adicional para una determinada emisión el proceso de filtro para la depuración de fallecidos lo considere para anular su orden de pago.	Ejecutar el filtro del proceso de depuración de fallecidos.	Emisión, proceso y número de ejecución	Se registra las cuentas que no tienen proceso de planilla normal y que están eliminadas por fallecimiento del pensionista.	Se consideró en el filtro la cuenta que solo tiene una planilla adicional.
2	4	Comprobar para el caso de una cuenta de pensión tenga para una emisión planillas tanto la normal como adicional que la anulación lo haga comenzando con la	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión y número de ejecución.	Se muestre el mensaje de que no es la orden de pago más reciente cuando la orden de pago no es la última ejecutada.	Mostró el mensaje que la orden de pago no es la más reciente.

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
		fecha más reciente.				
3	4	Validar que la orden de pago se anule y en el log de errores salga el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago" si se detecta que se realizó una modificación o anulación posterior a la orden de pago.	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión y número de ejecución.	Se muestre un mensaje si el último saldo de la cuenta esta anulado.	Mostró el mensaje cuando el saldo a revertir está anulado.
4	4	Verificar que la anulación del registro de saldos sea de la cuenta-prestación-ley-concepto del proceso y ejecución asociado a la orden de pago.	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión y número de ejecución.	La reversión se haga por cuenta, prestación, ley, proceso y ejecución.	Se revirtió el pago del devengado de la historia de saldos en base a la cuenta, prestación, ley, proceso y ejecución.
5	4	Validar la reversión de la historia de saldos y de la historia fraccionada.	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión y número de ejecución.	La reversión se haga en el detalle de la historia fraccionada.	Se revirtió el pago del devengado en el detalle de la historia fraccionada.
6	4	Comprobar que para el caso de consignaciones la anulación sea para el proceso y ejecución correspondiente y que la consignación se marque como intervenida.	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión, código de consignatario y número de ejecución.	Cuando existen dos pagos de consignaciones con diferentes procesos se anule el pago del consignatario cuyo proceso se ha elegido a anular.	Se anuló el pago de consignación con el código de proceso escogido.
1	5	Validar para una anulación de una orden de pago puntual o masivo que me permita anular la orden de	Ejecutar el proceso de depuración de fallecidos.	Proceso, cuenta de pensión y número de ejecución.	Se anula todas las órdenes de pago comenzando con la más reciente.	Se anuló todas las órdenes de pago comenzando por la más

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
		pago más reciente.				reciente.
2	5	Comprobar para el caso de una anulación masiva si en el archivo de carga envío el código de proceso como XXX el proceso anula todas las órdenes de pago asociada a la cuenta en una emisión determinada.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión con sus respectivos procesos.	Se anula todas las órdenes de pago para la cuenta cuando en el archivo se coloque XXX en vez del código de proceso.	Se anuló todas las órdenes de la cuenta de pago cuando se coloca los caracteres XXX.
3	5	Validar que no se anula la orden de pago si existe alguna posterior.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión con sus respectivos procesos.	Se graba en el archivo log si existe una cuenta cuyo proceso no es la más reciente.	Se grabó cuentas en el archivo log cuyo proceso no correspondió con la más reciente.
4	5	Validar que se anule la orden de pago y que en el log de errores salga el mensaje "Existe registro de movimiento de saldo posterior a la orden de pago hacer la reversión de saldos manualmente" si se detecta que se realizó una modificación o anulación posterior a la orden de pago.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión con sus respectivos procesos.	Se muestra un mensaje en el archivo log cuando se detecta que existe un saldo que fue anulado.	Mostró un mensaje en el archivo log con el mensaje que existe un registro de movimiento saldo posterior a la orden de pago.
5	5	Verificar que la anulación del registro de saldos sea de la cuenta-prestación-ley-concepto del proceso y ejecución asociado a la orden	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión	La reversión del saldo se hace por cuenta, prestación, ley, proceso y ejecución.	Se revirtió el saldo por cuenta de pensión, prestación, ley, proceso y ejecución.

Número de caso de prueba	Número de historia de usuario	Propósito	Actividad		Resultados	
			Inicialización	Descripción de los datos de entrada	Esperados	Reales
		de pago.		con sus respectivos procesos.		
6	5	Validar la reversión de la historia de saldos y de la historia fraccionada.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión con sus respectivos procesos.	La reversión de del pago de devengado se hizo para aquellas cuentas que tienen saldos.	Se revirtió el pago de devengados para las cuentas que tuvieron saldos.
7	5	Comprobar que se haya actualizado con el estado AC-Activo si el estado anterior fue PA-Pagado.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Cuenta de pensión y proceso o un Archivo con la información de cuentas de pensión con sus respectivos procesos.	El estado de la historia fraccionada que se encontraba en PA – Pagado se cambia a AC – Activo.	El estado de la historia fraccionada que se encontraba en PA – Pagado se cambió a AC – Activo.
8	5	Comprobar que para el caso de consignaciones la anulación sea para el proceso y ejecución correspondiente y que la consignación se marque como intervenida.	Ejecutar el proceso de anulación de orden de pago (individual o masivo).	Proceso, cuenta de pensión, código de consignatario y número de ejecución.	Se intervenga el pago del consignatario.	Se intervino el pago del consignatario.

5.3.4.7 PRIMERA ITERACIÓN

Se llevaron a cabo los dos primeros ítems de los requerimientos priorizados; se realizó el análisis de los estudios empíricos de aplicación de metodologías de desarrollo ágil.

Se finalizó el diseño y configuración del entorno de la ejecución de la metodología propuesta.

Se efectuó una búsqueda de herramientas que diesen soporte al desarrollo ágil.

Se analizó las amenazas que pudiesen poner en peligro la validación la ejecución de la metodología propuesta. Adicionalmente se realizó un plan de métricas que permitiese alcanzar los objetivos planteados.

En la etapa de elaboración, se definió la lista de requerimientos priorizados, se realizó una primera definición sencilla y clara de las características que debía cumplir el sistema.

Los requerimientos priorizados, artefacto que contiene las historias de usuario a alto nivel que guiaron el proceso de desarrollo y que posteriormente, a medida que avanzó las iteraciones, fueron descompuestas en un nivel mayor de detalle.

Comenzó con la reunión de planificación. La comunicación con el cliente, la disponibilidad completa del cliente que habitualmente existió al desarrollar un proyecto real.

La representación de la funcionalidad a través de historias de usuario y la estimación del tiempo necesario para efectuar cada tarea. Hubo falta de conocimiento de los desarrolladores en el producto al inicio de las iteraciones.

Existieron dificultades en diseño que no fueron consideradas en la planificación de la iteración propició que comenzasen a aparecer retrasos.

Factor de Desarrollo	Valor
Duración	Número total de días hábiles: 7
Objetivo marcado	• Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado. • Registro del proceso y ejecución por el proceso de carga de cuenta corriente.
Desarrollo	• Número de objetos a modificar: 5 • Líneas de código: 50

Factor de Desarrollo	Valor
Objetivos obtenidos	25%
Puntuación retrospectiva	-22[49]

Tabla 5.8: Tabla resumen iteración 1.

5.3.4.8 SEGUNDA ITERACIÓN

En la reunión de planificación se prestó especial atención en definir las historias usuario en un nivel mayor de detalle para realizar una planificación realista de acuerdo a los recursos disponibles. El objetivo propuesto para esta iteración fue completar las historias de usuario.

Dichas historias de usuario fueron descompuestas en un nivel mayor de detalle.

Finalmente, la historia de la segunda iteración quedó constituida por un subconjunto de historias de usuario.

El trabajo en esta iteración avanzó a buen ritmo ajustándose de un modo más aproximado a la planificación planteada. El diseño, implementación y pruebas se sucedía de forma ágil, en colaboración con el cliente.

La técnica de programación en pareja solo se aplicó según era necesaria.

El diseño de pruebas se hizo de forma simultánea al diseño e implementación. Se realizaron reuniones en las que cada miembro del equipo debía describir el estado del trabajo que tenía designado, lo que provocó una estimación imprecisa.

Cambiar de hábitos es siempre una tarea complicada. El aumento de la interacción entre el grupo de trabajo y el contacto continuo con el cliente favoreció la comunicación y solución de problemas.

La motivación en el trabajo y la práctica de programación en pares, resultó positiva. Hubo limitaciones técnicas en la parte de pruebas.

---

[49] Ver Anexo – Resultados de Retrospectiva.

La motivación y comunicación entre el grupo de trabajo se vio igualmente muy reforzada. Los aciertos en la fase de planificación de la iteración, produciéndose un gran avance en la definición de las historias de usuario.

No obstante, el equipo obtuvo una conclusión positiva de este hecho pues comprobó como la metodología se adaptaba a los cambios imprevistos del cliente para evolucionar el producto.

Factor de Desarrollo	Valor
Duración	Número total de días hábiles: 20
Objetivo marcado	• Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de tutor. • Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de riesgo. • Registro del proceso y ejecución mediante la aprobación del motivo de solicitud modificación y anulación de saldos. • Registro del proceso y ejecución mediante la aprobación del motivo de regularización de pensiones. • Registro del proceso y ejecución mediante la aprobación del motivo de unificación de cuentas.
Desarrollo	• Número de objetos: 8 • Líneas de código: 100
%Objetivos obtenidos	80%
Puntuación retrospectiva	-30

Tabla 5.9: Tabla resumen iteración 2

5.3.4.9 TERCERA ITERACIÓN

El cliente se centra en tratar aspectos sobre lo que necesita que haga el sistema pero no tiene la visión suficiente sobre otros aspectos críticos.

El aumento de conocimiento en el sistema que propició que se comprobara que muchas de los evolucionados en la iteración pasada fuesen erróneas, es decir, no se necesitaba evolucionar tanto código para conseguir la funcionalidad.

La reunión retrospectiva se constató que gracias al buen análisis que se realizó del punto de partida, la planificación fue más realista.

Factor de Desarrollo	Valor
Duración	Número total de días hábiles: 7
Objetivo marcado	• Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor. • Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.
Desarrollo	• Número de objetos: 9 • Líneas de código: 80
%Objetivos obtenidos	50%
Puntuación retrospectiva	-4

Tabla 5.10: Tabla resumen iteración 3.

5.3.4.10 CUARTA ITERACIÓN

A pesar del desbordamiento de trabajo, no se optó por incorporar nuevos componentes al equipo. Se consideró el comenzar la implementación de la cuarta iteración.

Hubo tareas que no se realizaron adecuadamente, principalmente en la parte de pruebas.

A pesar de esta situación, los objetivos primarios alcanzados para esta iteración fueron excelentes.

Factor de Desarrollo	Valor
Duración	Número total de días hábiles: 16
Objetivo marcado	• Modificación del Filtro. • Anulación de la Orden de Pago. • Revertir Saldo y Cuota de Devengado Planilla Normal incluye la ley 27803. • Revertir Saldo y Cuota de Devengado Planilla Adicionales. • Anulación de Consignaciones por

Factor de Desarrollo	Valor
	medio de la opción de depuración de fallecidos y anulación de órdenes de pago puntual y masivo. • Anulación de consignaciones mediante la opción del módulo de consignaciones.
Desarrollo	• Número de objetos a modificar: 5 • Líneas de código: 120
Objetivos obtenidos	125%
Puntuación retrospectiva	11

5.3.4.11 QUINTA ITERACIÓN

Debido al amplio conocimiento que proporcionaba el informe obtenido en las iteraciones anteriores en el que se empleó la alta comunicación, se avanzó más rápido.

Se llevó a cabo una refactorización del mismo para mejorar, limpiar y hacer más legible el código.

La interacción entre el grupo facilitó a la última iteración de la ejecución de la metodología. El equipo tenía pleno conocimiento tanto del dominio de aplicación como de los productos que estaban siendo evolucionados por lo que el trabajo en esta iteración fue sumamente ágil.

A mitad de la iteración las labores en el producto ya se daban por concluidas, lo cual se pudo incrementar el número de pruebas.

La satisfacción de éste fue elevada y el grupo creado al utilizar la metodología propuesta resultó muy positivo.

El equipo logró adaptarse a las variabilidades de las necesidades del cliente debido a la división del trabajo en ciclos de corta duración y a la estrecha interacción con el mismo.

La ágil redistribución del trabajo en situaciones críticas, permitió que los resultados obtenidos desde la perspectiva industrial fuesen muy satisfactorios tanto para el cliente como para el propio equipo. Para algunos miembros del equipo la continua asistencia a distintos tipos de reuniones, la

dificultad de disponer del cliente in situ durante el 100% del proceso de desarrollo del perjudicó el avance del producto.

Factor de Desarrollo	Valor
Duración	Número total de días hábiles: 19
Objetivo marcado	Anulación de Órdenes de Pago Puntual. Anulación de órdenes de Pago Masiva. Revertir saldo y cuota de devengado. Anulación de Consignaciones.
Desarrollo	• Número de objetos a modificar: 6 • Líneas de código: 150
Objetivos obtenidos	150%
Puntuación retrospectiva	38

CAPÍTULO VI

ANÁLISIS DE LOS RESULTADOS Y EVALUACIÓN DE HIPÓTESIS

6.1 CARACTERÍSTICAS DEL PRODUCTO DURANTE LA EVOLUCIÓN

En esta sección se analiza la evolución de los productos. Se ha tratado de un proceso de integración continua a pequeña escala, a través del cual, y mediante historias de usuario, se han ido sucediendo incorporaciones de funcionalidad en el entorno de producción.

La figura 6.1 muestra gráficamente la evolución de los requerimientos priorizados que guiaba el desarrollo, agrupando las historias de usuario tanto para el entorno de construcción, integración y pruebas.

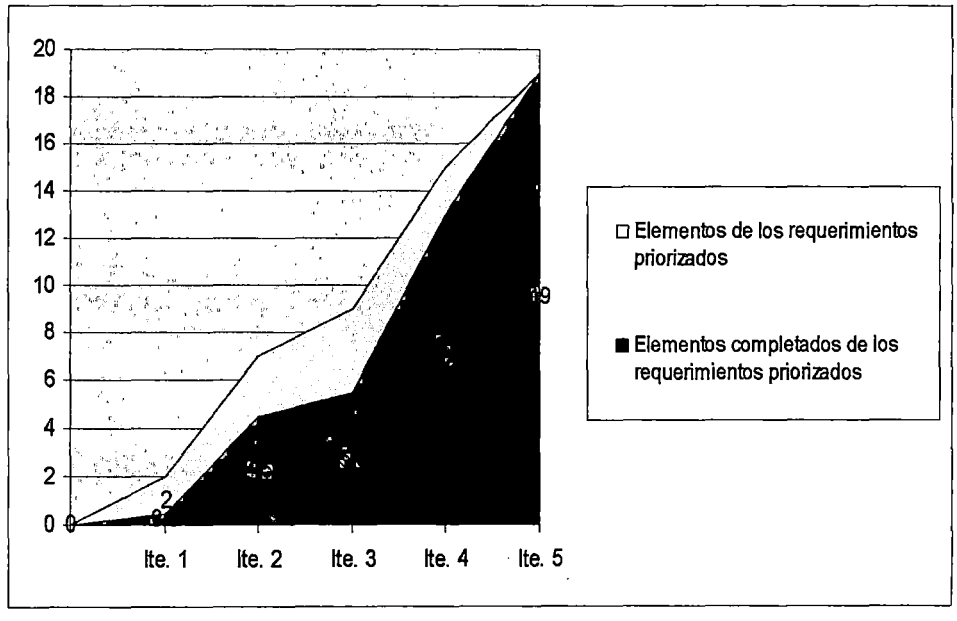


Figura 6.1: Evolución del requerimiento priorizados a lo largo del desarrollo

El área celeste refleja el incremento de funcionalidad que sufre el producto en su evolución, entendiendo por producto el conjunto de todos, en las sucesivas iteraciones. Durante el desarrollo se pasa una lista de requerimientos priorizados constituido por apenas 2 historias de usuario en la iteración 1 hasta alcanzar una dimensión de 19 historias de usuario en la última iteración del proyecto. El hecho de que a medida que avanzaba el proyecto, el nivel de detalle con el que se veían los componentes fuese mayor influyó decisivamente en este sentido. Pues las 2 historias de usuario iniciales, referentes a cada uno de los componentes, fueron descompuestas a lo largo del desarrollo en función de los atributos de dichos componentes. De esta forma, y en continua interacción con el cliente, se fue refinando el producto.

El área lila muestra la integración de funcionalidad que ha ido sufriendo el producto a lo largo del desarrollo, conforme se daban por concluidas las historias de usuario. Puede deducirse que a medida que avanzaban las iteraciones, aumentaba, casi exponencialmente, la funcionalidad completada. En la figura 6.2, que muestra el porcentaje de necesidades del cliente cubiertas en cada iteración respecto al conjunto total de necesidades, puede observarse con mayor detalle esta circunstancia. El producto fue desarrollado completamente con un porcentaje final de realización de los requerimientos priorizados de un 100%.

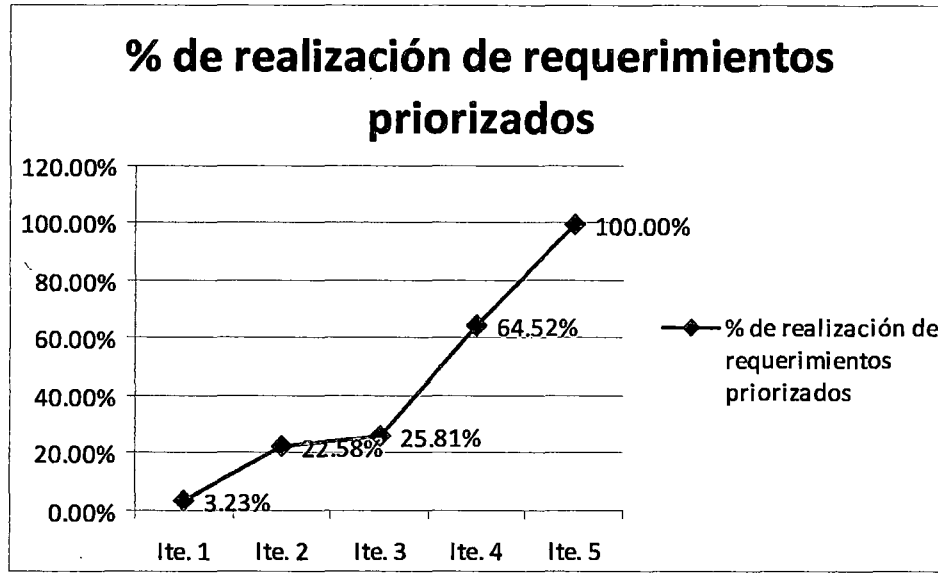


Figura 6.2: Evolución del porcentaje de realización de los requerimientos priorizados.

Cabe destacar que el proceso de integración a gran escala que se realiza en las metodologías convencionales puede resultar incluso más complejo que la propia codificación. En el caso concreto, la ejecución de la metodología propuesta, al tratarse de integraciones mucho más pequeñas [varias veces, o como mínimo una, al día], esta práctica no ha supuesto grandes problemas ya que la evolución del producto se ha realizado progresivamente, considerando la continua interacción con el cliente muy positiva en este sentido. La clave para que la integración continua haya sido viable fue disponer de una batería de pruebas, de tal forma que una vez que el nuevo código estaba integrado con el resto, se ejecutaba toda la batería de pruebas para validar la integración.

En la figura 6.3 se grafica la diferencias de avances de tal manera que se llega a cumplir con lo planificado. Con estos resultados se validó que la nueva metodología permite construir o modificar el producto software en su totalidad[50].

[50] La metodología no hace diferencia entre las actividades de construcción del desarrollo y el mantenimiento, ya que en ambos se realizan las mismas actividades.

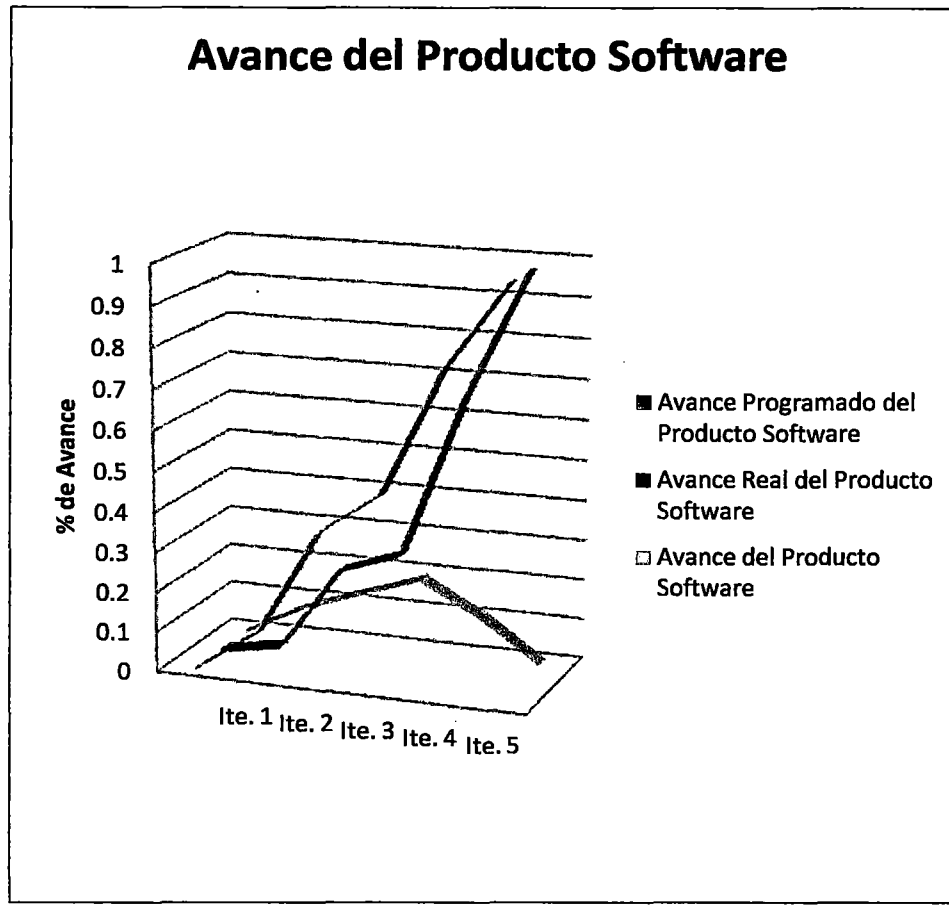


Figura 6.3: Avance del Producto Software. Elaboración Propia.

La evolución en líneas de código y para el producto, puede verse en la figura 6.4, respectivamente. Del mismo modo, la figura 6.5 muestra la evolución en los objetos impactados durante el desarrollo. Cabe destacar, que la evolución en número de líneas de código del producto apenas es palpable, pues al tratarse de un producto de evolución, las líneas nuevas se ven compensadas con las líneas eliminadas. No obstante, el producto final es de un tamaño sensiblemente menor que el producto de partida. Sin embargo, el impacto en términos de objetos es elevado. Puede observarse cómo los objetos impactados aumentan al principio del desarrollo. La principal explicación que se deduce de esta circunstancia es el hecho de que al comienzo del desarrollo, la falta de conocimiento en el producto a evolucionar, influye en que se manipulen más objetos de las que, en

en realidad, es necesario manipular pues no se tiene el conocimiento suficiente para hacer esta tarea. A medida que este conocimiento aumenta se conocen los objetos implicados en la evolución del producto por lo que el impacto es menor.

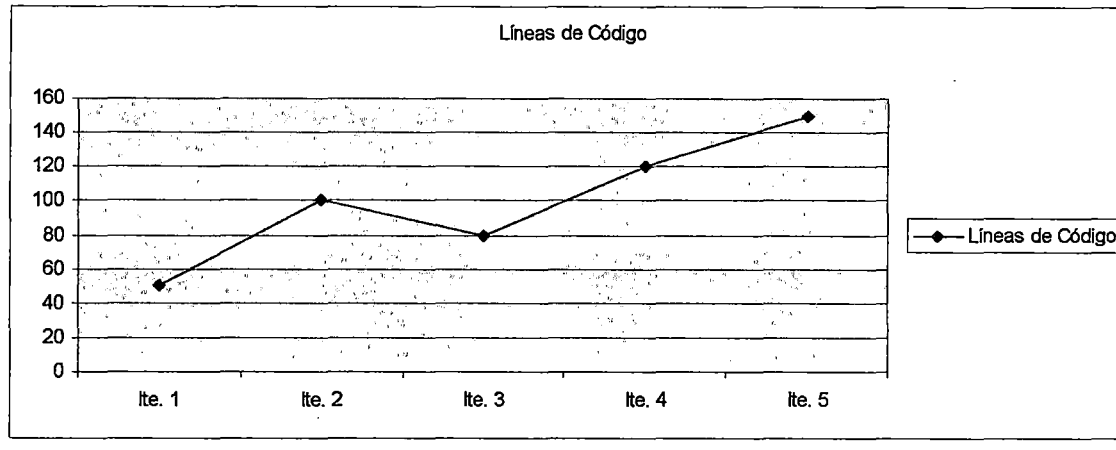


Figura 6.4: Evolución del impacto en las líneas de código del producto software

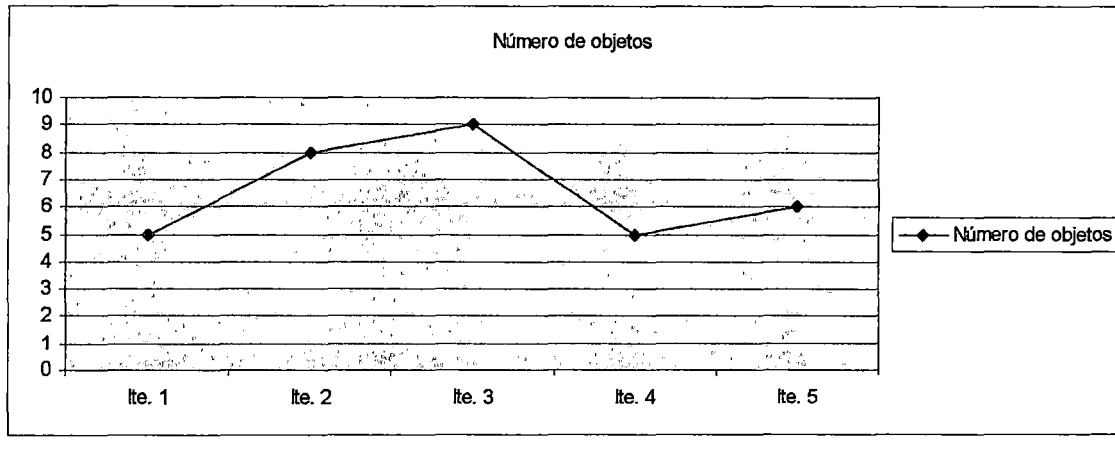


Figura 6.5: Evolución del impacto en los objetos del producto de software.

Otro aspecto destacable en este sentido es el esfuerzo dedicado al proyecto para evolucionar el producto a lo largo de las iteraciones.

Se puede concluir, que en el desarrollo de esta ejecución de la metodología, la experiencia adquirida por el equipo durante el desarrollo, tanto en la metodología como en el dominio de aplicación del producto, ha sido crucial en el volumen de funcionalidad integrada. De este modo, la

funcionalidad integrada desde la segunda iteración hasta el final de proyecto es muy superior a la integrada en ciclos anteriores cuando aún no se tenía un conocimiento profundo en el producto a evolucionar y no se estaba habituado a las directrices establecidas por la metodología propuesta.

6.2 ASPECTOS DE AGILIDAD EN EL DESARROLLO

Se analizó la evolución de la productividad en las distintas iteraciones que se han desarrollado hasta conseguir el producto final, con el objetivo de dar respuesta a una de las principales cuestiones planteadas, referente a cómo evoluciona la agilidad en el desarrollo al utilizar una metodología ágil como la propuesta.

Para el estudio de la productividad, ésta se ha medido en términos de objetivos alcanzados de los requerimientos priorizados final, en cada iteración, respecto al esfuerzo invertido en el mismo. En primer lugar, la figura 6.6 visualiza gráficamente el esfuerzo destinado al proyecto en cada iteración, medido en horas.

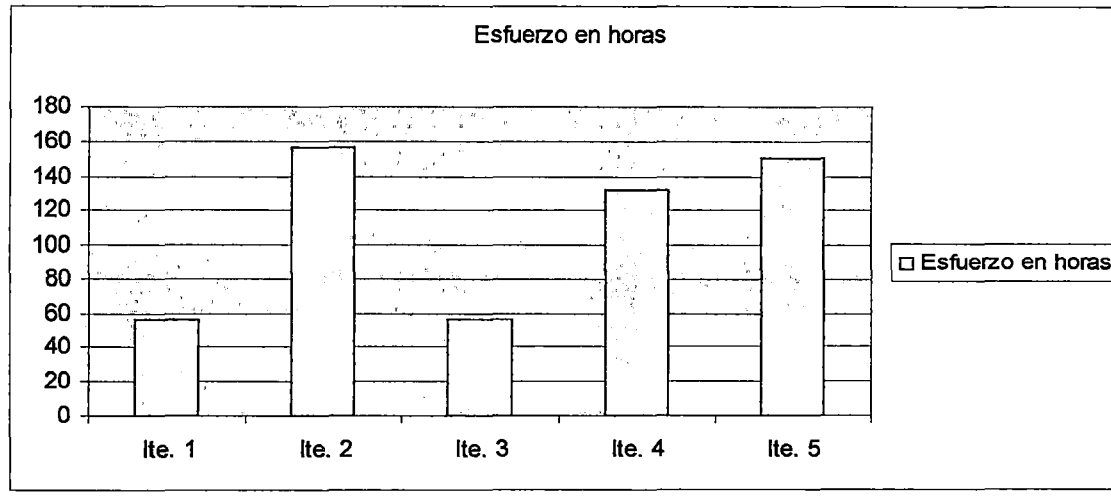


Figura 6.6: Esfuerzo dedicado al proyecto en cada iteración

El descenso de motivación a mediado del proyecto junto con la aparición de un conjunto de acontecimientos propios del contexto de profesionales en el que se desarrolla el experimento pero ajenos al mismo, propició la imposibilidad para algunos miembros del equipo de cumplir su

compromiso y, consecuentemente, el esfuerzo dedicado en esta parte del desarrollo es sensiblemente menor. Por otro lado, la agilidad alcanzada en la última iteración explica que se cumpliesen los objetivos del mismo con una destinación de recursos menor.

Una vez conocidos los valores de esfuerzo dedicado al proyecto, y considerando el porcentaje de realización de los requerimientos priorizados en cada iteración que aparece en la figura 6.2, se ha calculado, en términos absolutos, el valor de la productividad en cada iteración del modo que se indica a continuación:

$$\text{Productividad [Sx]} = \left(\frac{\%Objetivos\_alcanzados\_requerimientos\_priorizados\,(ite.x)}{Esfuerzo\_dedicado(ite.x)}\right)*100$$

La figura 6.6 muestra los resultados obtenidos.

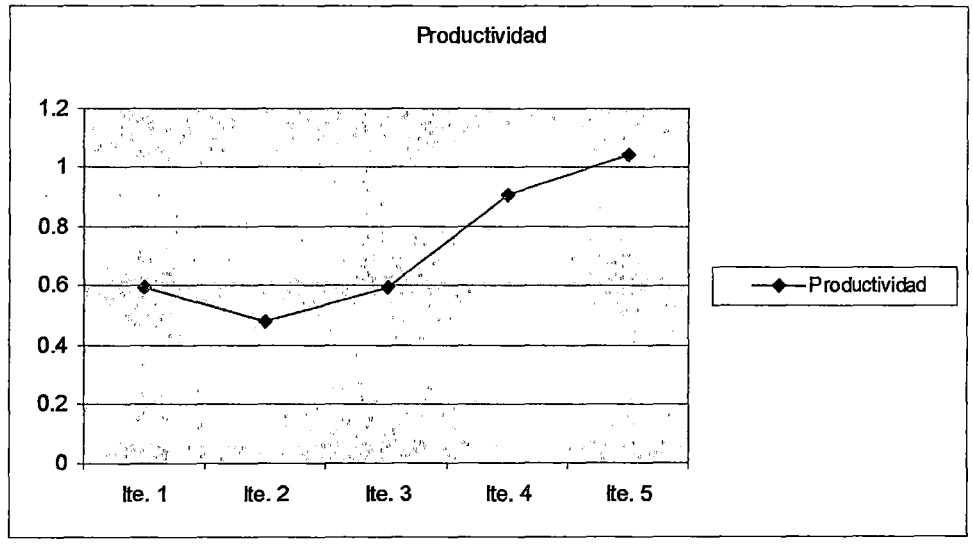


Figura 6.7: Evolución de la productividad a lo largo del proyecto.

Los valores de la productividad están íntimamente ligados con la evolución del producto descrita en el apartado anterior. Como muestra la gráfica, a medida que transcurren las iteraciones aumenta la productividad en el proceso, es decir, a medida que transcurren las iteraciones se cumplen más objetivos y se es más ágil. Existen dos factores principales que explican esta situación:

- Por un lado, la adaptación a la metodología en las primeras iteraciones repercute negativamente en el tiempo dedicado a cumplir los objetivos en sí de la iteración, puesto que gran parte del esfuerzo se destina a conocer la metodología, habituarse a ella y aprender a utilizar las herramientas consideradas en este entorno. Además, se debe considerar que este proceso de adaptación repercute negativamente en que las tareas establecidas por la metodología se realicen de forma incorrecta minando la productividad del desarrollo. Así, los errores por ejemplo en la etapa de planificación influyen decisivamente en que el porcentaje de objetivos cubiertos en esta parte sea muy pequeño.
- Por otro lado, la ausencia de un conocimiento profundo del dominio de aplicación ha influido drásticamente en que las historias de usuario aparezcan a un nivel muy alto de detalle y no se puedan completar. Además, la falta de conocimiento en el producto a evolucionar hace que en las primeras iteraciones sea difícil evolucionar notablemente el producto porque los desarrolladores deben destinar un alto porcentaje del tiempo a conocer, en cierta medida, la arquitectura y el código del producto antes de proceder a su evolución.

La figura 6.7 muestra gráficamente la evolución en el porcentaje de historias de usuario completadas por iteración, considerando en este caso la historia de las iteraciones. Se puede observar que a medida que avanza el proyecto las planificaciones de las iteraciones son más certeras, gracias al buen análisis que se realiza del punto de partida, aumentando claramente el porcentaje de cubrimiento de las historias de las iteraciones. Por tanto, partir de una situación real y planificar de forma no optimista y reflexionando todos los miembros del equipo ha sido fundamental para realizar buenas planificaciones.

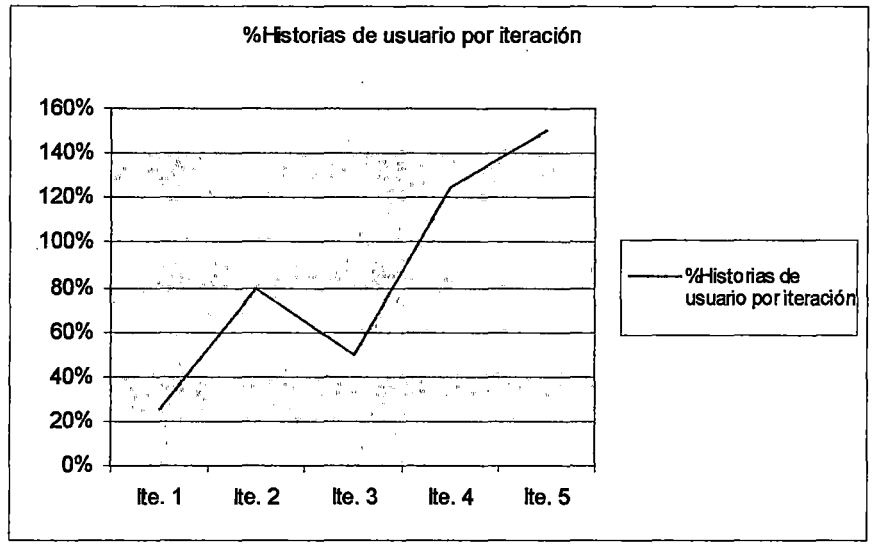


Figura 6.8: Evolución de las historias de usuario completadas por iteración

En líneas generales el cumplimiento de objetivos en las iteraciones ha sido incremental. Evidentemente, la evolución de un producto software incorpora un conjunto de dificultades no consideradas en productos de nueva construcción.

Por otro lado, es interesante analizar el esfuerzo que se ha destinado a las distintas tareas para lograr estos valores de productividad y explicar por qué ha evolucionado de este modo. La figura 6.8 muestra la distribución del esfuerzo destinado al proyecto para cada una de las tareas consideradas [RUP, SCRUM y XP].

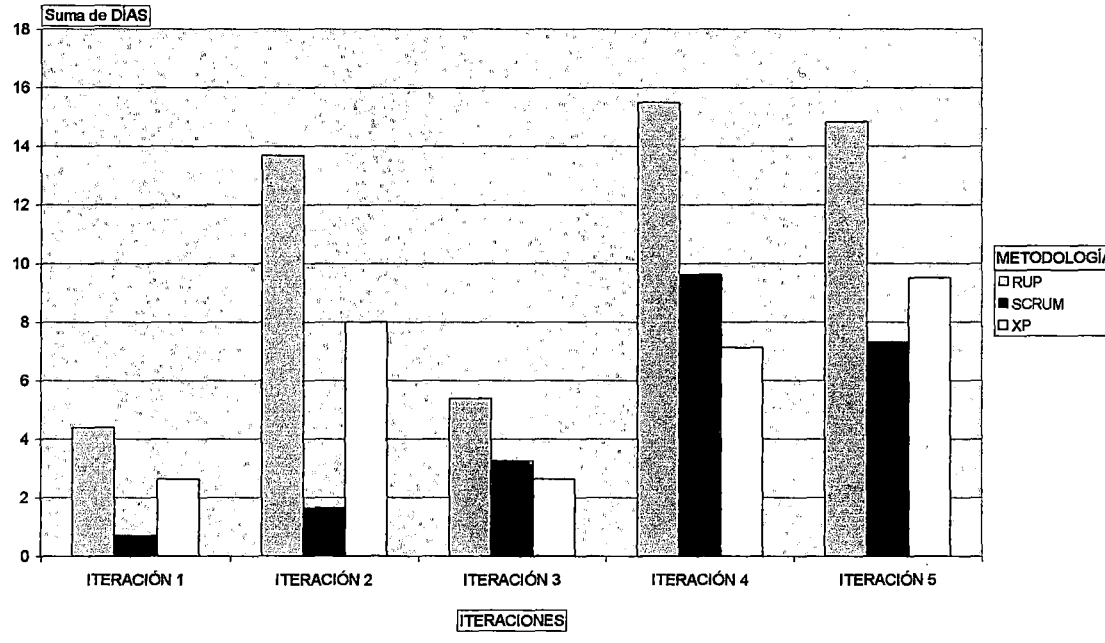


Figura 6.9: Distribución de esfuerzos a lo largo del desarrollo

Como puede observarse, los esfuerzos en las primeras iteraciones se encontraban muy distribuidos entre las distintas tareas. El tiempo dedicado a tareas ágiles relacionadas con la metodología, a la definición y gestión de las historias de usuario y al diseño es muy elevado, en detrimento del esfuerzo dedicado a tareas de integración continua [codificación y pruebas] lo que redunda altamente en el incumplimiento de objetivos en sí de cada iteración. A medida que se fue adquiriendo conocimiento en la metodología de desarrollo y en el dominio de aplicación los esfuerzos en codificación pasaron a copar el más alto porcentaje, y con el ellos el cumplimiento de objetivos. Una reflexión que cabe hacerse en este sentido es si el hecho de realizar una etapa tan ligera en la definición de las necesidades del sistema influya en que en las primeras iteraciones no se tenga el conocimiento suficiente para cumplir los objetivos previstos.

La figura 6.10 muestra gráficamente la distribución global de esfuerzos que se ha realizado en el proyecto.

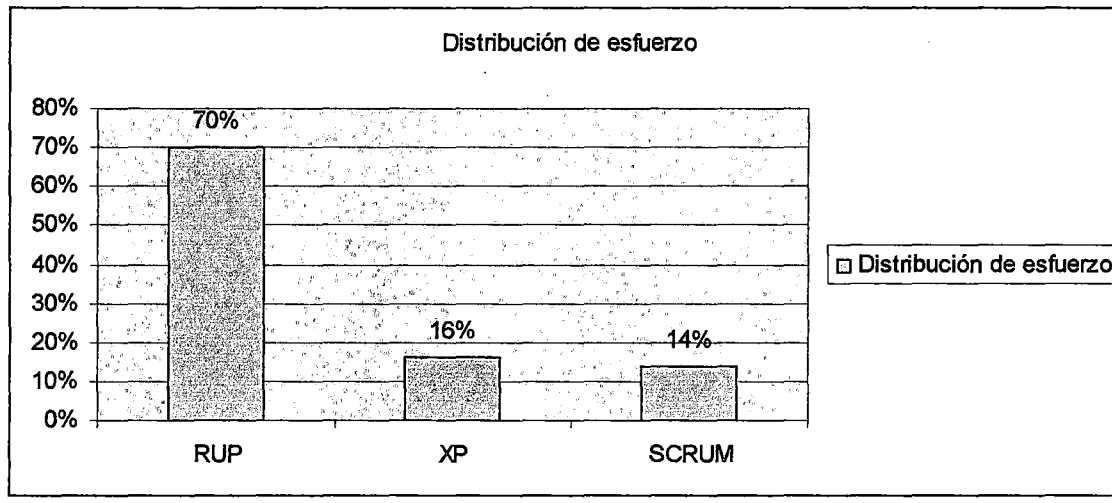


Figura 6.10: Distribución global de esfuerzos

Se puede apreciar que realizar el producto de software se está utilizando metodologías ágiles el cual permiten la realización del producto con las características especificadas del cliente.

6.3 CALIDAD DEL PRODUCTO OBTENIDO

Uno de los aspectos que más satisfacción ha aportado tanto al cliente como al equipo de desarrollo es la calidad del producto obtenido.

El hecho de que el producto esté especialmente diseñado para ser adaptable a diferentes dominios de aplicación ha repercutido muy positivamente en este sentido, pues se conocía de antemano las partes variantes del producto que, por tanto, debían ser probadas tras el incremento de funcionalidad. En un principio una parte del equipo se dedicó a realizar la validación de este producto. No obstante, al constatar que no se encontraban fallos en el producto pues los aspectos a probar resultaban tan obvios que ya habían sido probados por los propios desarrolladores, se decidió que fuesen los propios desarrolladores quienes ejecutasen este trabajo. Aunque parezca una contradicción, pues se entiende que los desarrolladores no deben probar sus propios productos, no se encontraron defectos en las sucesivas entregas de este producto al cliente. Esta es una de las principales ventajas de la evolución de productos software, ya que las

tareas de pruebas se simplifican enormemente al saber de antemano que partes del producto son susceptibles de ser probadas.

La tasa de defectos encontrados en la fase de pruebas sigue la distribución que aparece en la figura 6.11

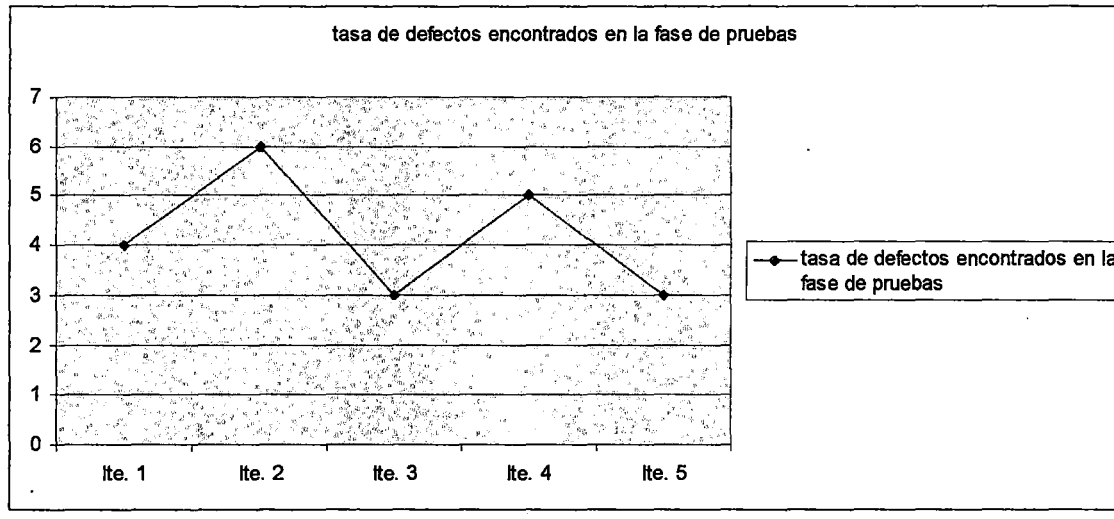


Figura 6.11: Evolución de la tasa de errores en el producto software.

El número de defectos encontrados es muy bajo ya que el hecho de que dos personas se encarguen de construir este producto simultáneamente favorece la obtención de productos de calidad.

6.4 EVALUACIÓN DE LAS HIPÓTESIS Y DISCUSIÓN

La hipótesis afirma la factibilidad de diseñar una nueva metodología de desarrollo y mantenimiento de software a fin que mejore la atención de requerimientos de los sistemas de información.

Es decir, se demuestra que es posible identificar las mejoras prácticas de las metodologías de desarrollo existentes que se adecúen a la fábrica de software.

Los objetivos de la metodología propuesta comprenden:

• La metodología debe permitir desarrollar sistemas de software con la calidad requerida y a satisfacción del usuario.

• La metodología debe permitir desarrollar gran variedad de sistemas de software.

• La metodología debe soportar todas las etapas de desarrollo de software.

• La metodología debe permitir superar o mitigar las limitaciones o desventajas de las metodologías de desarrollo de software tradicional, en lo referente a facilitar el desarrollo y mantenimiento de los sistemas software.

El logro de los objetivos de la metodología se ha podido lograr a través de los siguientes componentes: definición de roles, fases e hitos, disciplinas dentro de las fases, disciplinas de soporte y artefactos. Asimismo, se implementó patrones de desarrollo basados en la máxima comunicación, comunicación interna y externa, participación del cliente. Finalmente, se incluyeron los componentes de la arquitectura ágil, integración continua y calidad en el proceso.

Se puede enumerar las limitaciones y las características mínimas que debe cumplir la nueva metodología.

Las buenas prácticas de las metodologías RUP, XP y SCRUM permiten asegurar la alta calidad de la producción de software [cronograma y presupuesto], mejorar la comunicación, incorporar alto nivel de disciplina de las personas que participan en el proyecto, manejar la impredictibilidad y el riesgo a niveles adaptables.

Se puede diseñar la metodología mediante la combinación de las metodologías de desarrollo como el Proceso Unificado de Desarrollo [RUP], y de metodologías ágiles [como el XP y SCRUM].

El total de tareas identificadas en la nueva metodología comprende 19 actividades. De las cuales, el porcentaje de aplicación correspondiente a las tareas de RUP, XP y SCRUM representan una participación adecuada de acuerdo a las necesidades del cliente del caso de estudio. [Ver anexo, para mayor detalle de las tareas de cada método de desarrollo].

Se puede aplicar la nueva metodología de desarrollo en la construcción de un producto de software.

En efecto, la evolución favorable del índice del avance del producto de software con respecto al avance programado del producto de software brinda evidencias de que la nueva metodología de desarrollo es aplicable para el caso de estudio desarrollado.

6.5 LIMITACIONES DE LA INVESTIGACIÓN

- No se profundiza el análisis de procesos de desarrollo. La principal necesidad que cubre la metodología, se enfoca en identificar factores [principios ágiles] para mejorar las aplicaciones.
- No se evalúa los costos incrementales de la aplicación de la nueva metodología. Sin embargo, cabe considerar que se cuenta con herramientas libres para la implementación de la metodología. [Ver acápite sobre las Buenas Prácticas y Adaptabilidad del RUP, SCRUM y XP]. Asimismo, se incide principalmente en reforzar las actividades de gestión del conocimiento y comunicación.
- No se ha elaborado un test estadístico de evaluación comparativa de la metodología actual de la fábrica de software con respecto a la nueva metodología. Este trasciende los

objetivos de esta tesis; sin embargo, el análisis del caso de estudio permite aceptar la hipótesis de investigación.

- Existen otros métodos ágiles que pueden servir de referencia para la nueva metodología de desarrollo de software. Al respecto, [Fernandez, 2010] desarrolla la especificación de la metodología ágil para desarrollo de software, apoyada en la herramienta Case Genexus.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES:

1. Este trabajo se ha desarrollado siguiendo los lineamientos de las metodologías SCRUM, *Rational Unified Process* [RUP] y *Extreme Programming* [XP].
2. Desde el punto de vista metodológico, el marco ágil del trabajo desarrollado, fue utilizado como base para la aplicación de una nueva metodología de desarrollo, demostrando de esta manera su amplia adecuación y alto grado de utilidad en el desarrollo de sistemas.
3. El sistema desarrollado como un prototipo de arquitectura, evidencia que el ciclo de vida del software basado en el marco ágil de la metodología propuesta, cumple con el objetivo de la metodología.
4. Con la aplicación de la metodología propuesta a una Entidad del Estado, se demuestra que la combinación de las metodologías de desarrollo de software SCRUM, XP y RUP, permiten desarrollar software con las características especificadas por el cliente.
5. Analizando los resultados del caso de estudio se desprende lo siguiente:

- **Agilidad en el desarrollo**: Las necesidades del cliente han sido cubiertas dentro del cronograma programado. Los resultados obtenidos respecto a la productividad del equipo han sido

satisfactorios. La productividad del equipo se ha incrementado casi linealmente a lo largo del desarrollo. El uso de metodologías ágiles en la evolución del producto ha propiciado que el desarrollo se caracterice por una mayor rapidez en la resolución de problemas y un excelente rendimiento del equipo de trabajo. Asimismo, factores como la cohesión e interacción entre el grupo de trabajo, la motivación en el desarrollo y la continua interacción con el cliente se han considerado muy positivos.

- **Calidad del producto obtenido**: El conocimiento que se tenía de la parte susceptible de modificación ha favorecido el proceso de pruebas obteniendo un producto sin errores. Asimismo, la calidad del producto obtenido ha sido también alta, favorecido, principalmente, por las prácticas de programación en pares y de refactorización utilizadas en este desarrollo.
- **Esfuerzo de adaptación a la metodología**: El esfuerzo inicial de adaptación a la metodología ha sido considerable, principalmente en la adquisición de nuevos hábitos. Durante el desarrollo, este proceso ha pasado por diferentes etapas, alternando puntos de mayor y menor esfuerzo. No obstante, al final del mismo se observó un alto grado de adaptación en la mayor parte del equipo.
- **Satisfacción del cliente**: La satisfacción del cliente ha ido incrementando a lo largo del desarrollo del sistema a través de la consecución incremental de objetivos en la evolución del producto, la continua aportación de valor a su negocio, al permitir de forma progresiva la ejecución de la funcionalidad del producto.
- **Satisfacción del equipo:** La satisfacción del equipo de desarrollo ha sufrido variaciones a lo largo del proceso causadas, principalmente, por aspectos dependientes del contexto universitario, ajenos al experimento. Se puede concluir

en este sentido, que utilizar las metodologías ágiles en proyectos paralelos resulta complicado pues merma la motivación y satisfacción de los involucrados en varios proyectos a la vez. No obstante, la reflexión final de la mayoría de los componentes del equipo es favorable a la metodología.

6. La falta de madurez de las metodologías ágiles implica que aún existan importantes lagunas en su aplicación y se hagan necesarias ciertas "correcciones". Este trabajo ha descrito diversos problemas encontrados en la evolución del producto con metodologías ágiles. La mayor parte de los problemas detectados se centran en la definición de las necesidades del cliente utilizando metodologías de desarrollo ágil. Este análisis ha sido posible gracias a que el producto objeto de la investigación es en realidad la evolución de otro existente, desarrollado utilizando metodologías convencionales y del que se disponía de una especificación de requisitos tradicional. El proceso de definición de las necesidades del cliente ha tenido algunas dificultades: dificultad de identificación de algunas necesidades, principalmente aquellas que se apartan del foco de la funcionalidad, necesidades del cliente que por su naturaleza afectan de forma transversal al proyecto [problema identificado como transversalidad], requisitos derivados, granularidad y documentación de algunas necesidades del usuario. Además de identificar y tipificar los problemas que pueden surgir, se aportan soluciones para que, siguiendo un desarrollo ágil, conseguir llegar a la meta a tiempo y en presupuesto. Respecto a los requisitos no funcionales, mientras la identificación de algunos se podría asociar a historias de usuario, no es el caso de otros cuyo estudio se debería abordar de forma explícita e independiente de las historias de usuario bajo el concepto propuesto de historia de sistema.

RECOMENDACIONES:

1. Las empresas al desarrollar proyectos de software deben tomar la metodología más adecuada, analizando las ventajas y desventajas que éstas puedan brindar al entorno del proyecto, tomando muy en cuenta las características propias del negocio, ya que la metodología será la que debe adaptarse y acoplarse con estas características, y no que la empresa se adapte a la metodología seleccionada.
2. Dado la mayor acogida que las metodologías ágiles tienen actualmente en el mercado, es muy aconsejable seleccionarlas para el desarrollo de proyectos de software, el alcance o tamaño del mismo no afectará en la eficiencia de la metodología, sino mas bien dependerá de como los miembros del grupo de trabajo logran el acoplamiento de la metodología y el proyecto.
3. SCRUM es una de las metodologías ágiles que se acopla fácilmente con la gestión de riesgos, esta propone una constante revisión del proyecto, realizando un control sutil del mismo, permitiendo detectar y solucionar inconvenientes de manera oportuna; por lo que se recomienda a SCRUM una opción muy viable al momento de seleccionar la metodología con la que se va a trabajar.
4. Se recomienda usar la propuesta sugerida para reducción de riesgos basada en la metodología SCRUM, de manera continua durante todas las iteraciones del desarrollo del sistema, de esta manera lograr que el grupo de trabajo se familiarice con la propuesta, y pueda ejecutarse con mayor fluidez y rapidez durante el proyecto, además hay que tomar en cuenta que esta puede ser aplicada sin importar el nivel del alcance del proyecto, ya que tanto la propuesta y la metodología pueden acoplarse a proyectos de grande o pequeño alcance.

GLOSARIO DE TÉRMINOS

- *Ágil*: Una manera iterativa e incremental de desarrollo de software que sigue el Manifiesto Ágil y sus principios relacionados.
- *Burndown, gráfico*: Es un gráfico que muestra la cantidad de trabajo restante en una iteración. El eje X indica el tiempo y el eje Y la cantidad de trabajo.
- *Consignatarios o Alimentistas*: Derecho de pensión que recibe un hijo de otro compromiso del pensionista.
- *Equipo de desarrollo*: Un grupo interdisciplinario de profesionales encargado del desarrollo del software.
- *Indicadores de resultados*: Miden el desempeño financiero de la organización.
- *Indicadores operativos*: Miden la eficiencia y el desempeño de las operaciones o procesos dentro de la organización.
- *Inventario [I]*: Todo el dinero invertido por el sistema en cosas a ser vendidas. En el sistema de desarrollo de software se entiende por las funcionalidades no cobradas al cliente y los costos totalmente variables en los que se ha incurrido.
- *Iteración*: Un periodo de tiempo determinado durante el cual el equipo planea, desarrolla, prueba y entrega un conjunto de funcionalidades.

- *Manifiesto Ágil*: Declaración de principios y valores que definen el desarrollo ágil de software.
- *Meta*: El objetivo para el cual el sistema fue creado.
- *Orden de Pago*: Documento que se genera para que los pensionistas puedan ejecutar su pago a través de la ventanilla de un banco. Allí se detallan los ingresos y egresos del pensionista.
- *Plan de proyecto*. Documento que describe el enfoque técnico y de gestión que seguirá un proyecto. Generalmente, el plan describe el trabajo a realizar, los recursos necesarios, los métodos a utilizar, los procesos a seguir, los programas a cumplir y la forma en la que se organiza el proyecto.
- *Prototipo*. Versión preliminar de un sistema que sirve de modelo para fases posteriores.
- *Prueba de sistema*. Prueba cuya finalidad es evaluar el grado de conformidad con los requisitos de un sistema completo.
- *SCRUM diario*: Una reunión corta al iniciar el día de trabajo donde el equipo sincroniza sus actividades y los avances realizados.
- *SCRUM Master*: persona responsable del proceso de Scrum y guía del equipo de desarrollo.
- *SCRUM*: Es una metodología Ágil desarrollada en los años 90 por Jeff Sutherland y Ken Schwaber, propone un desarrollo incremental de un producto por medio de iteraciones.
- *Iteración*: SCRUM utiliza el término iteración para referirse a cada iteración del proyecto.
- *Throughput*: Es la tasa a la que se genera efectivo a través de la entrega de software funcional. Sólo cuando el cliente ha realizado el pago por los entregables se ha generado Throughput.

- *Unidades de desarrollo*: Definen el nivel de complejidad de una función de desarrollo y de una manera aproximada las horas de trabajo necesarias para concluirla.

BIBLIOGRAFÍA

[Arroba, 2011] Arroba Medina, Lilian Elizabeth, *Propuesta de Aplicación de SCRUM para Minimizar los Riesgos en un Proyecto de Desarrollo de Software*, Tesis de grado para la obtención del título de ingeniero en sistemas informáticos y de computación, Quito - Ecuador, Escuela Politécnica Nacional – Facultad de Ingeniería de Sistemas, 2011, 111 páginas.

[Becerra, 2002] Becerra Guzmán, Sammyr Alejandro, *Desarrollo de un Sistema de Vigilancia Corporativo Compatible con Dispositivos de Telefonía Móvil*, Proyecto previo a la obtención del título de ingeniero en sistemas informáticos y de computación, Quito, Escuela Politécnica Nacional – Facultad de Ingeniería de Sistemas, 2009, 120 páginas.

[Beck, 2001] K. Beck. Manifesto for agile software development. Technical Report, 2001. http://www.agilemanifesto.org/.

[Beck, 2002] Beck, Kent, *Extreme Programming Explained*, Second Edition, United States of America, Addison Wesley Professional, 2004, 224 pages.

[Boehm,1988] Barry W. Boehm: A Spiral Model of Software Development andEnhancement. IEEE Computer, 21, 5 (May 1988), pp. 61-72.

[Boehm et al 2000] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald Reifer, Bert Steece: Software Cost Estimation with COCOMO II, Upper

Saddle River, New Jersey: Prentice-Hall PTR, 2000, xxxviii+502 pp.; ISBN: 0-13-822122-7.

[Cabrero, 2009] Cabrero Moreno, Daniel, *Construcción y Evolución del Software Basados en Valor*, Tesis Doctoral, España, Universidad de Castilla-La Mancha - Departamento de Tecnologías y Sistemas de la Información, 2009, 260 páginas.

[Capiluppi, 2007] Capiluppi, A., Fernandez-Ramil, J., Higman, J., Sharp, H.C., Smith, N.: An empirical study of the evolution of an agile-developed software system. In: 29th International Conference on Software Engineering [ICSE], pp. 511-518. [2007].

[Cobb, 2011] Cobb, C. [2011]. *Making Sense of Agile Project Management Balancing Control and Agility*. John Wiley & Sons, Inc. Hoboken.

[Cohen, 2010] Cohen, G. [2010]. *Agile Excellence for Product Managers a Guide to Creating Winning Products with Agile Development Teams*. Super Star Press. Silicon Valley.

[Cockburn 2000] Cockburn, Alistair, *Agile Software Development*, Third Edition, Cockburn – Highsmith Series Editors, 2000, 220 pages.

[Cruz, 2010] Cruz Sandoval, Dagoverto, *Herramienta de Soporte a la Valorización rápida se procesos software utilizando el modelo de Moprosoft Bajo un Enfoque RIA*, Tesis para optar el título de ingeniero en computación, Huajuapan De León – México, Universidad Tecnológica de la Mixteca, 2010, 142 páginas.

[Dalcher, 2005] Dalcher, D., Benediktsson, O., y Thorbergsson, H., "Development Life Cycle Management: A Multiproject Experiment", Proceedings of the 12th International Conference and Workshops on the Engineering of Computer Based Systems [ECBS'05], 2005.

[Dapena, 1999] Rodriguez-Dapena, P., "Software safety certification: a multidomain problem," Software, IEEE , vol.16, no.4, pp.31-38, Jul/Aug 1999.

[Díaz, 2008] J. Díaz, Agustín Yagüe, Pedro P. Alarcón and Juan Garbajosa. "A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools". Aceptado en 7th IEEE International Conference

on Composition-Based Software Systems [ICCBSS 2008].Madrid. Spain. February 2008.

[Dyba, 2008] Dyba, T., Dingsoyr, T.: Empirical Studies of Agile Software Development: A Systematic Review, Information and Software Technology doi: 10.1016/j.infsof.2008.01.006 [2008].

[Fowler, 2005] M. Fowler. The new methodology. Technical report, 2005. http://www.martinfowler.com/articles/newMethodology.html.

[Fernández, 2010] Fernández Verástegui, Daniel Luis, *Especificación de Metodología Ágil para Desarrollo de Software, apoyada en la herramienta Case GENEXUS*, Tesis para optar el grado académico de Maestro en Ciencias con mención de Sistemas, Lima – Perú, Universidad Nacional de Ingeniería, Facultad de Ingeniería Industrial y de Sistemas - Sección de Posgrado, 2010, 278 páginas.

[Guibovich, 2011] Guibovich Arroyo, Rosario Elizabeth, Montes Gutierrez, Marleni, Paredes Arteaga, Juan Carlos, Pari Guerrero, Katia Lourdes, *Modelo de Procesos Enfocado al Desarrollo y Mantenimiento de Software Diseñado para la Software Factory De GMD S.A.*, Programa De Alta Especialización PAE - Universidad Esan, Lima - Perú, 2011, 249 páginas.

[Guzmán, 2008] Guzmán Ávila, Andrés Alejandro, *Desarrollo de un Sistema de Puntos de Ventas para Micromercados, Utilizando la Metodología Extreme Programming*, Tesis de Grado de Ingeniero de Sistemas e Informática, Sangolquí - Ecuador, Departamento de Ciencias de la Computación – Escuela Politécnica el Ejército, 2008, 156 páginas.

[Highsmith, 2004] Highsmith, Jim, *Agile Project Management: creating innovate products,* First Edition, United States of America, Addison Wesley, 2004, 277 pages.

[Kassab, 2007] Kassab, M., Daneva, M., Ormandjieva, O.: Scope Management of Non-Functional Requirements. In: EUROMICRO '07: Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp.409—417. Washington [2007].

[Keith, 2010] Keith, C. [2010]. *Agile Game Development with Scrum.* Addison-Wesley Professional. Indiana.

[Kitchenham, 2002] Kitchenham, B.A.; S. L. Pfleeger; L. M. Pickard; P. W. Jones; D. C. Hoaglin; K. El Emam; and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," IEEE Transactions on Software Engineering, vol. 28, No. 8, pp. 721-733, 2002.

[Kniberg, 2010] Kniberg, H., Skarin M. [2010]. *Kanban y Scrum – Obteniendo lo mejor de ambos.* C4Media. USA.

[Leffingwell, 2007] Leffingwell, D. [2007]. Scaling Software Agility: Best Practices for Large Enterprises. Addison Wesley. Boston.

[Lehman, 1997] Lehman, M.M. [1997] Laws of Software Evolution Revisited, pos. pap., EWSPT96, Oct. 1996, LNCS 1149, Springer Verlag, 1997, pp. 108-124

[Mann, 2005] Mann, C., Maurer, F.: A case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In: Proceedings of the Agile Development Conference [ADC'05]. IEEE Computer Society [2005].

[Marcos, 1998] Marcos, Esperanza, *Investigación en Ingeniería del Software vs. Desarrollo Software*, España, Universidad Rey Juan Carlos, 1998, 16 páginas.

[Mitaritonna, 2010] Mitaritonna, Alejandro Daniel, *CAPA-3: Una innovadora metodología para el desarrollo de software en ambientes de trabajo virtuales*, Tesis de Maestría en Ingeniería en Sistemas de Información, Ciudad Autónoma de Buenos Aires - Argentina, Facultad Regional Buenos Aires - Universidad Tecnológica Nacional, 2010, 166 páginas.

[Núñez, 2010] Núñez Mori, Jose Germán, *Usabilidad En Metodologías Ágiles*, Tesis de Master en Ingeniería del Software, Madrid – España, Universidad Politécnica de Madrid Facultad de Informática, 2010, 237 páginas.

[Palacios, 2010] Palacios Álvarez, Nicanor, *La Teoría de Restricciones Aplicada al Desarrollo de Software,* Tesis para la obtención de grado de magíster, Ecuador, Universidad Andina Simón Bolívar Sede Ecuador - Área

de Gestión Programa de Maestría en Dirección de Empresas, 2010, 77 páginas.

[Pelayo, 2007] Pelayo García-Bustelo, Begoña Cristina, *Desarrollo ágil de Software con Arquitecturas Dirigidas por Modelos,* Tesis Doctoral, Oviedo – España, Universidad de Oviedo – Departamento de Informática, 2007, 415 páginas.

[Piattini, 2003] Piattini Velthuis, Mario G., Calvo-Manzano Villalón, José A., Cervera Bravo, Joaquín y Fernández Sanz, Luis, *Análisis y Diseño de Aplicaciones Informáticas de Gestión: Una Perspectiva de Ingeniería del Software*, Primera Edición, Madrid - España, RA-MA EDITORIAL, 2003, 736 páginas.

[Piattini, 2007] Piattini Velthuis, Mario y Garzás Parra, Javier, *Fábricas de software: experiencias, tecnologías y organización*, Primera Edición, México, Alfaomega Grupo Editor, S.A. de C.V., 2007, 560 páginas.

[Pichler, 2010] Pichler, R. [2010]. *Agile Product Management with Scrum Creating Products that Customers Love.* Addison-Wesley Professional. Stoughton.

[Pries, 2011] Pries, K., Quigley, J. [2011]. Scrum Project Management. CRC Press. Boca Raton.

[Rodríguez, 2008] Rodríguez González, Pilar, *Estudio de la Aplicación de Metodologías Ágiles para la Evolución de Productos Software,* Tesis de Máster en Tecnologías de la Información, España, Facultad de Informática - Universidad Politécnica de Madrid, 2008, 146 páginas.

[Rueda, 2010] Rueda Gutiérrez, Allan Balam, *Propuesta de una Guía para Interpretar los Procesos de MOPROSOFT de la Categoría de Operación Usando una Combinación de Métodos Ágiles*, Tesis de Grado de Maestro en Ciencias en Informática, México D.F., Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas- Instituto Politécnico Nacional, 2010, 217 páginas.

[Schenone, 2004] Schenone Marcelo, Hernán, *Diseño de una Metodología Ágil de Desarrollo de Software*, Tesis de Grado en Ingeniería en

Informática, Buenos Aires, Argentina, Facultad de Ingeniería - Universidad de Buenos Aires, 2004, 200 páginas.

[Schiel, 2010] Schiel, J. [2010]. *Enterprise-Scale Agile Software Development*. CRC Press. Boca Raton.

[Stamelos, 2007] Stamelos, I. Sfetsos, P. [2007]. *Agile Software Development Quality Assurance*. Information Science Reference. Hershey.

[Wellington, 2005] Wellington, A., Briggs, T., y Girard, C.D., "Comparison of Student Experiences with Plan-Driven and Agile Methodologies", Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, 2005.

[Wysocki, 2006] Wysocki, R. [2006]. *Effective Software Project Management*. Wiley Publishing, Inc. Indianapolis.

[Yagüe, 2008] Rodriguez, P., Yagüe, A., Alarcón, P.P., Garbajosa, J., "Metodologías ágiles desde la perspectiva de la especificación de requisitos funcionales y no funcionales". 13th Conference on Software Engineering and Databases [JISBD'08].

ANEXO 1

DEFINICIÓN DE MEDICIONES

Se ha tomado como referencia a [Rodríguez, 2008], donde presenta el plan de mediciones desarrollado para *evaluar la aplicación de la metodología propuesta*, concretamente, veremos los datos que serán recogidos como parte del estudio y la metodología seguida para su recolección. El objetivo principal es disponer de datos objetivos y consistentes que nos permitan obtener conclusiones basadas en hechos rigurosos y no en simples intuiciones u observaciones subjetivas. Dada la naturaleza de la investigación:

- Uno de los criterios fundamentales que se han considerado en el diseño del plan de mediciones es que el proceso de recogida de las mismas sea lo más ligero posible, de tal forma que no obstruya, o impacte en la menor medida posible, las actividades diarias del equipo ágil.

Las entidades de interés para dar respuesta a las cuestiones planteadas como objetivos la ejecución de la metodología propuesta y que, por tanto, van a ser medidas en él se pueden clasificar en tres grupos:

- *Las entradas o recursos* con los que se dota el desarrollo la ejecución de la metodología propuesta que, evidentemente, influirán en los resultados obtenidos en el mismo.
- *Los productos obtenidos*, concretamente la evolución de los productos en los diferentes iteración [sprint]s de los que consta

el experimento, a medida que se consolida la aplicación de la metodología, y la calidad de estos productos obtenidos.

- *Las actividades que forman el proceso de desarrollo la ejecución de la metodología propuesta y la productividad del mismo.*

Para el análisis de dichas entidades, los datos serán recogidos de tres fuentes de información:

- Registros con información automatizada,
- Cuestionarios y
- Observaciones en el lugar de trabajo.

EXPERIENCIA DEL EQUIPO

Al comienzo de la ejecución de la metodología se medirá el conocimiento y experiencia de cada uno de los miembros del equipo en los siguientes términos: grado académico, experiencia en desarrollo software, experiencia en gestión de proyectos, experiencia en el lenguaje de programación y experiencia en desarrollo ágil y programación en parejas. Además, dado que estamos evolucionando un producto ya existente será de interés medir

- El grado de conocimiento del producto y
- El grado de conocimiento del dominio al que se pretende adaptar dicho producto.

Dichas mediciones se tomarán también, si se diese el caso, en la incorporación de nuevos miembros al equipo.

ESFUERZO DEDICADO AL PROYECTO

Es interesante registrar las horas, o en su defecto minutos, que se dedican a realizar cualquier actividad relacionada con la ***adaptación al dominio***. Entre otras utilidades, estas mediciones servirán para analizar los objetivos obtenidos a lo largo del proceso respecto al ***esfuerzo dedicado*** al mismo.

Además permitirá recapacitar sobre el proceso de aprendizaje de la metodología y la posible influencia del esfuerzo dedicado a cada aspecto de

la misma. Esta medición está dirigida a todos los miembros del equipo, incluido el cliente. Para facilitar la recogida de la misma se utilizará la opción Hoja de Presencia de la herramienta Time Report[51] al finalizar cada tarea. En ella se deberá indicar la siguiente información:

- Seleccionar el Identificador del proyecto, por ejemplo: P-2008-0002: Iteración 4
- Indicar el Tipo de la tarea, ya que para facilitar la posterior gestión, las mediciones de tiempo se dividirán en tareas
- Fecha de recogida de la métrica
- Hora de comienzo de la tarea
- Duración [medido en minutos u horas si es un valor exacto]
- Notas [descripción de la tarea, si fuese necesario]

DEFECTOS EN CADA ITERACIÓN

Al finalizar cada iteración, con el objetivo de evaluar la calidad del producto obtenido, se medirán el número de defectos del mismo, así como la importancia y el origen de estos defectos. Esta métrica será recogida por la persona del equipo encargada de realizar las pruebas a través de una hoja Excel diseñada con el fin de recoger estas mediciones. En esta hoja deberá indicar la fecha de recogida de la medida, el identificador de la iteración así como el identificador del defecto encontrado, una descripción del defecto y una estimación de su importancia [alta, media o baja] y la causa dónde tuvo origen.

DEFECTOS EN PRODUCTOS ENTREGADOS

Asimismo, es de interés conocer los ***defectos encontrados por el cliente*** cuando el producto ya ha sido entregado. La continua interacción con el cliente facilitará la toma de esta medición. La información a recoger será similar a la de la métrica anterior, utilizando de nuevo la hoja Excel. Sin

[51] Herramienta utilizada por la fábrica de software.

embargo, en esta ocasión el encargado de toma la medida será el cliente en colaboración con el Dueño del Producto.

NÚMERO DE LÍNEAS DE CÓDIGO DEL PRODUCTO POR ITERACIÓN

Para evaluar la evolución del producto, al finalizar cada iteración se anotarán el número de líneas de código del producto construido hasta ese momento. De esta forma, se podría analizar incluso la evolución del producto cada día.

OBJETOS IMPACTADOS POR CADA HISTORIA DE USUARIO

Con propósito similar al de la métrica anterior, se van a medir las clases que sufren algún tipo de impacto al evolucionar el producto. Esta medición se tomará en función de la historia de usuario que este siendo desarrollada. El desarrollador, indicará el identificador de la iteración, de la historia de usuario y de la clase que está siendo impactada, especificando, asimismo, el tipo de impacto sufrido [objeto nuevo, objeto modificado o objeto eliminado] y la fecha de recogida de la medida.

PORCENTAJE DE REALIZACIÓN DE LOS REQUERIMIENTOS PRIORIZADOS

Para medir la productividad al utilizar una metodología de desarrollo ágil, se tomarán diferentes mediciones del porcentaje de realización del producto. El porcentaje de realización de los ***Requerimientos Priorizados*** será una de ellas. Esta medida será tomada por el dueño del producto en colaboración con el Coordinador del Proyecto al finalizar cada iteración y al finalizar el proyecto. Este porcentaje será calculado como se muestra a continuación:

- Elemento del requerimiento priorizado [BE]
- Número de requerimientos priorizados realizados [ABE]
- Número de elementos totales de los requerimientos priorizados [TBE]

- % Realización Requerimientos. Priorizados = $\frac{(\sum_{I=0}^{ABE} BEi * pi) * 100}{(\sum_{I=0}^{TBE} BEi * pi)}$
  - *pi*: peso asignado al elemento de los Requerimientos Priorizados ya que puede constituir un objetivo primario o secundario

Los datos serán recogidos en las hojas Excel diseñadas para tomar las medidas.

PORCENTAJE DE REALIZACIÓN DEL OBJETIVO DE LA ITERACIÓN

También se recogerá el porcentaje de realización del objetivo marcado para cada iteración. La medición se tomará de forma similar a la anterior, siendo la fórmula de cálculo la que aparece a continuación:

- Historia de usuario de la iteración [US]
- Número de historias de usuario totales [TUS]
- Número de historias de usuario completadas [CUS]
  - % realización de la iteración = $\frac{(\sum_{I=0}^{CUS} USi * pi) * 100}{(\sum_{I=0}^{TUS} USi * pi)}$
  - *pi*: peso asignado a la historia de usuario dentro de la iteración ya que puede ser un primaria o secundaria

PORCENTAJE DE REALIZACIÓN DE CADA HISTORIA DE USUARIO POR ITERACIÓN

Para que el análisis sea más completo, profundizando en un nivel mayor de detalle en la productividad de cada iteración, se medirá el porcentaje de realización de cada historia de usuario en el iteración en el que se define, así como, si no está completa en esa iteración, en iteraciones posteriores.

- Número total de tareas que componen la historia de usuario [TT]
- Número de tareas completas [CT]

- % realización de una historia de usuario = $\frac{CT*100}{TT}$

NÚMERO DE ITERACIONES HASTA COMPLETAR UNA HISTORIA DE USUARIO

Dado que puede suceder que las historias de usuario no se completen en la iteración planificada sino que aparezcan retrasos, en alineación con la medición anterior, se medirán estos retrasos contabilizando el número de iteración que se dedican a cada historia de usuario ya que, teóricamente, cada historia de usuario debería ser definida de tal forma que fuese capaz de desarrollarse en un único iteración. Esta medida será recogida al final del proyecto por el dueño del producto en las plantillas Excel habilitadas para las mediciones, indicando el identificador de la iteración en el que se incorpora la historia de usuario y el identificador de la iteración en el que finalmente es aceptada por el cliente.

HISTORIAS DE USUARIO REPRIORIZADAS

Uno de los aspectos más interesantes de las metodologías ágiles es su capacidad de adaptación a los cambios que se produzcan en el entorno. Por este motivo será interesante medir los cambios acontecidos a través de las historias de usuario que sean repriorizadas en el proyecto. Esta medida la tomará el dueño del producto a través de la plantilla Excel.

SATISFACCIÓN EN CADA ITERACIÓN

Finalmente, puesto que las metodologías ágiles están enfocadas a una nueva filosofía de desarrollo, resulta muy interesante conocer la opinión, percepción y observaciones tanto de los componentes del equipo como del cliente con esta nueva forma de trabajo. Así al finalizar cada iteración deberán rellenar un cuestionario acerca de estos aspectos en los que deberán realizar una breve descripción con reflexiones personales sobre la metodología seguida y valorar su grado de satisfacción [valor comprendido entre 0, satisfacción mínima, y 10, satisfacción máxima]. Al finalizar el

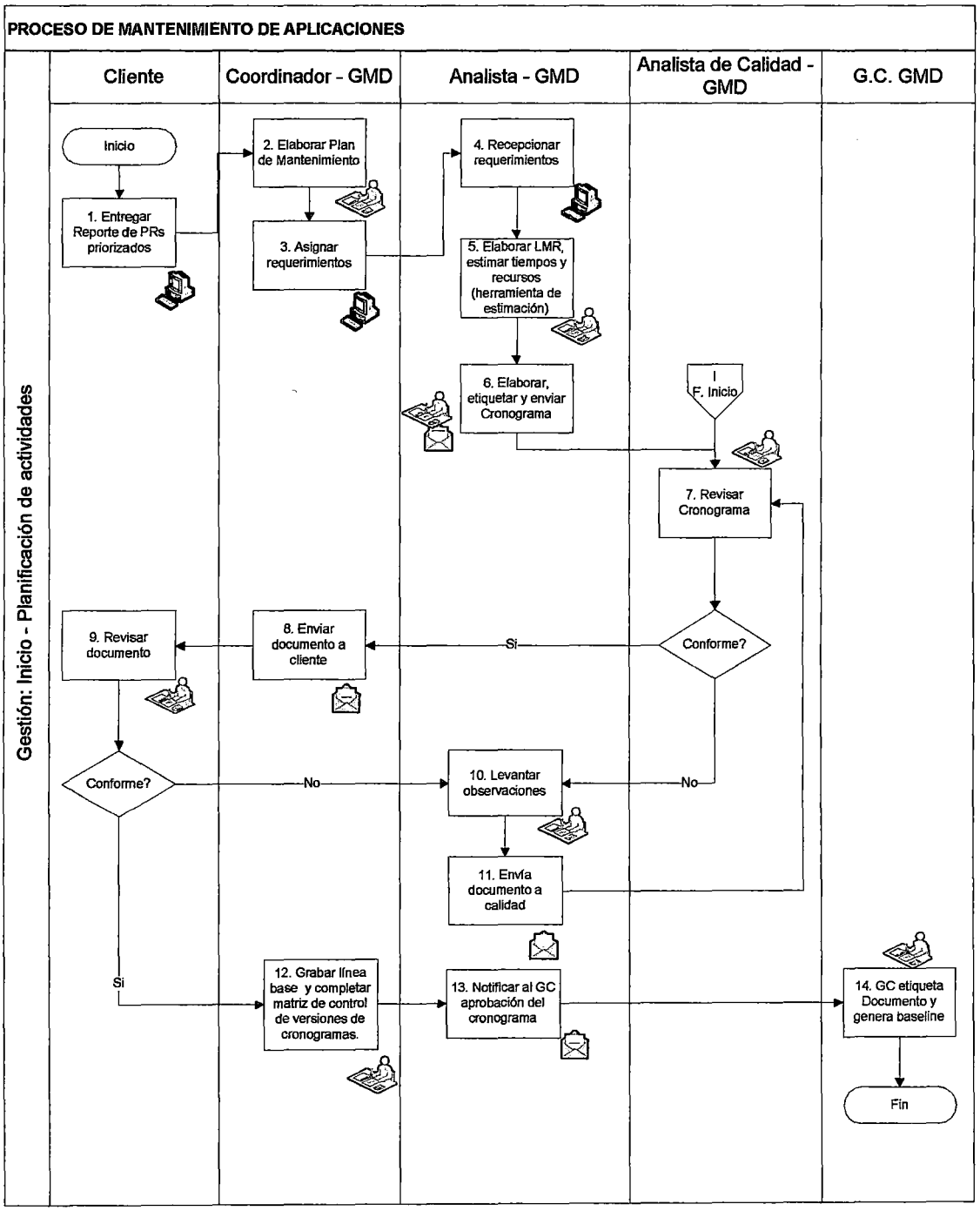
proyecto, del mismo modo, se rellenará un cuestionario similar con la satisfacción global de la metodología.

ANEXO 2

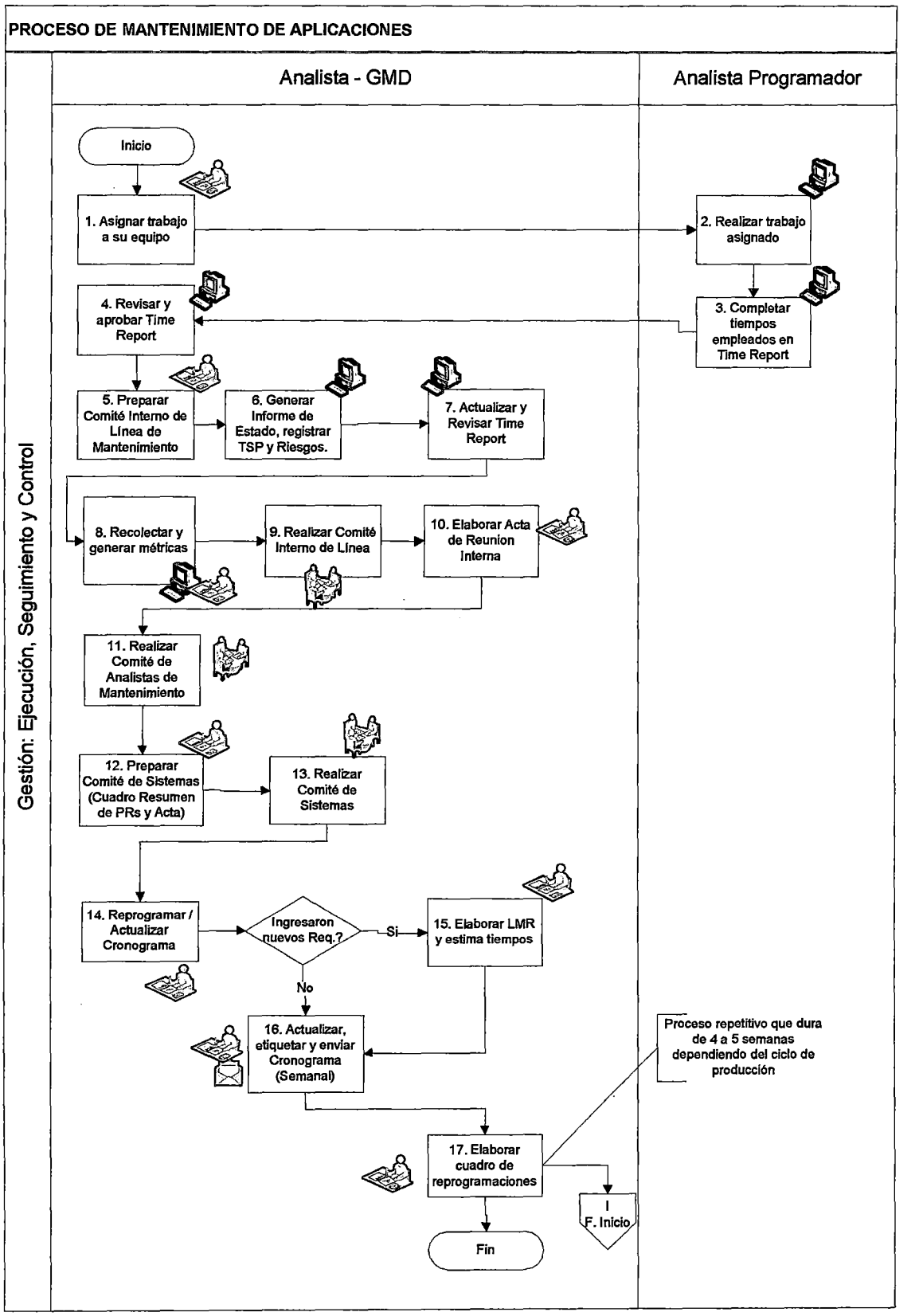
DIAGRAMA DEL PROCESO DE MANTENIMIENTO DE SOFTWARE

1. Diagrama del Proceso de Gestión de Mantenimiento

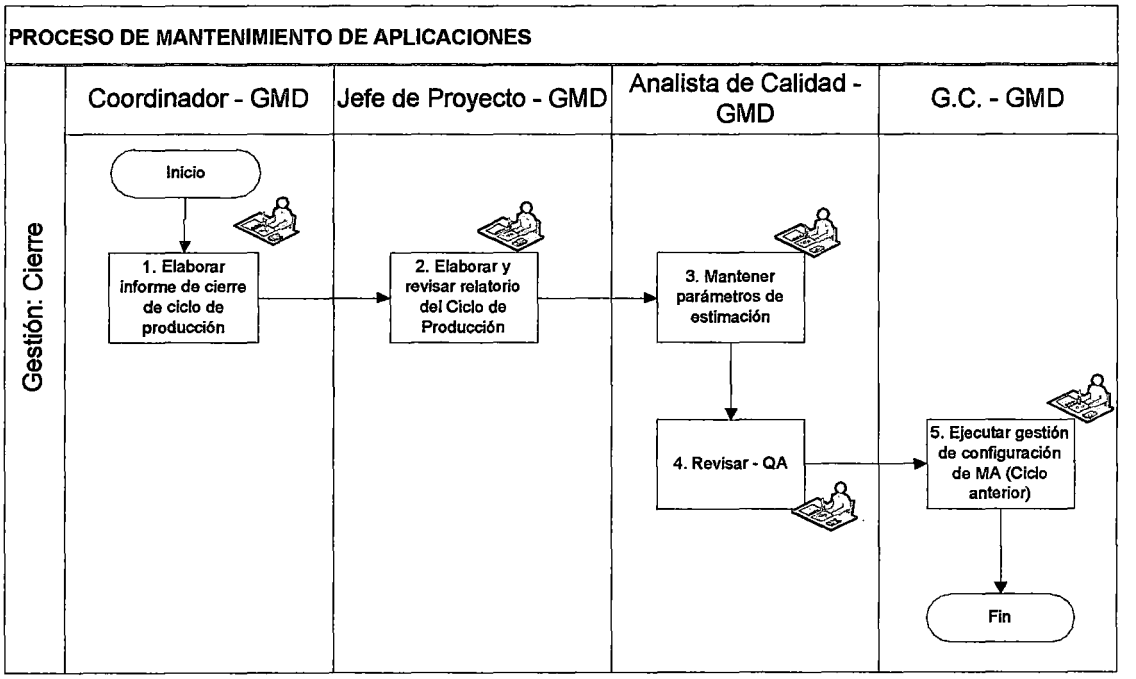
A. Fase de Inicio y Planificación



B. Fase de Ejecución Seguimiento y Control

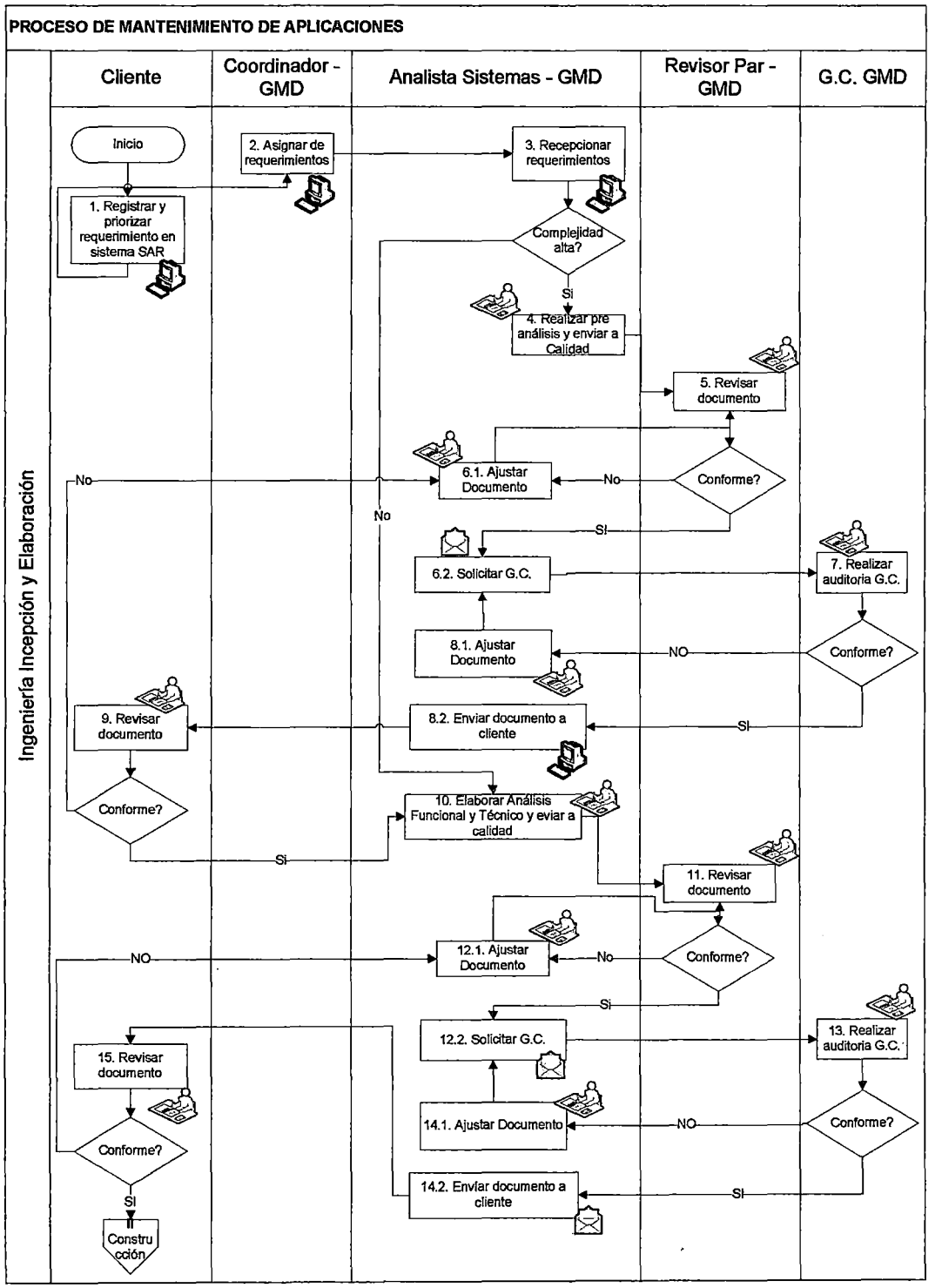


C. Fase de Cierre

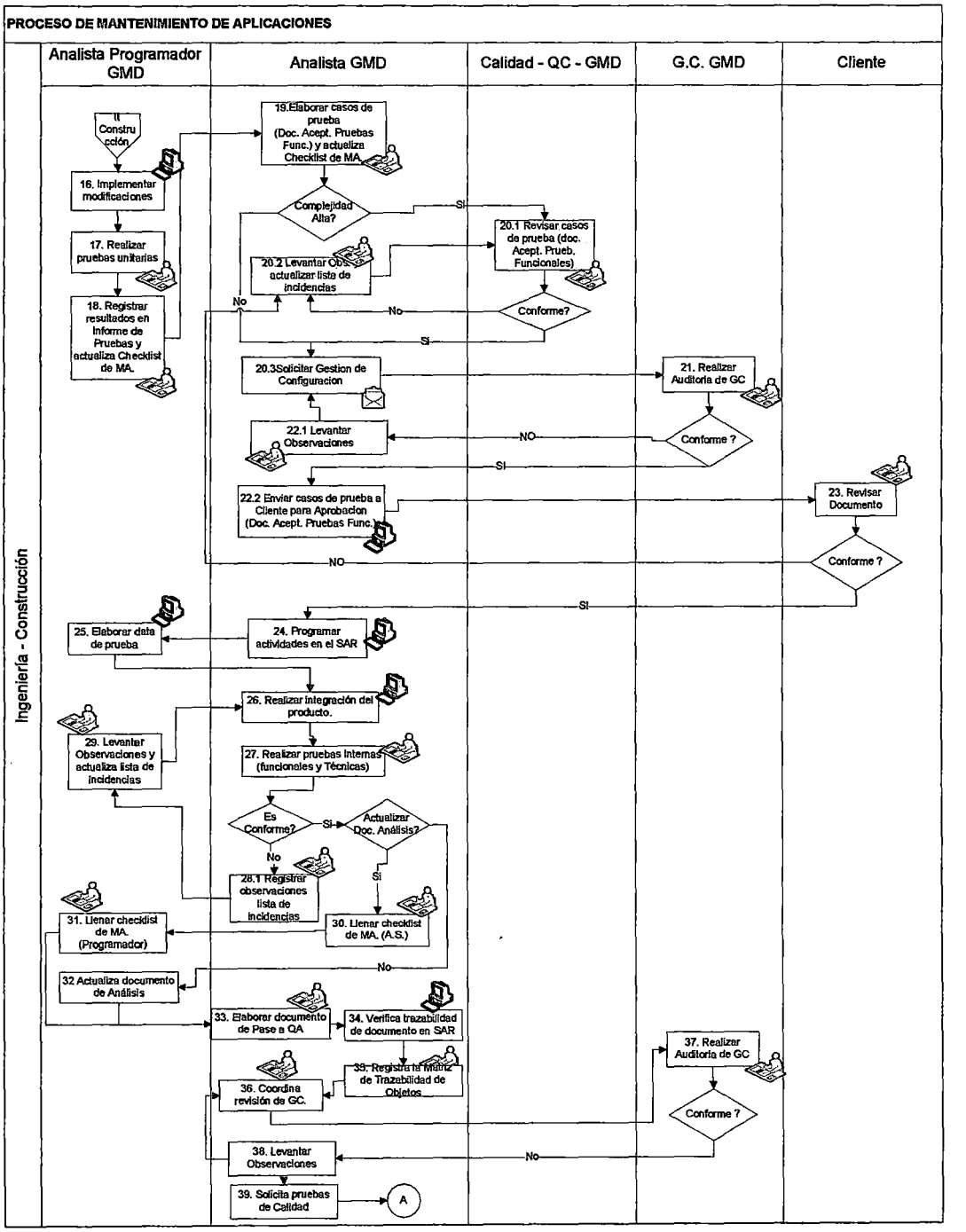


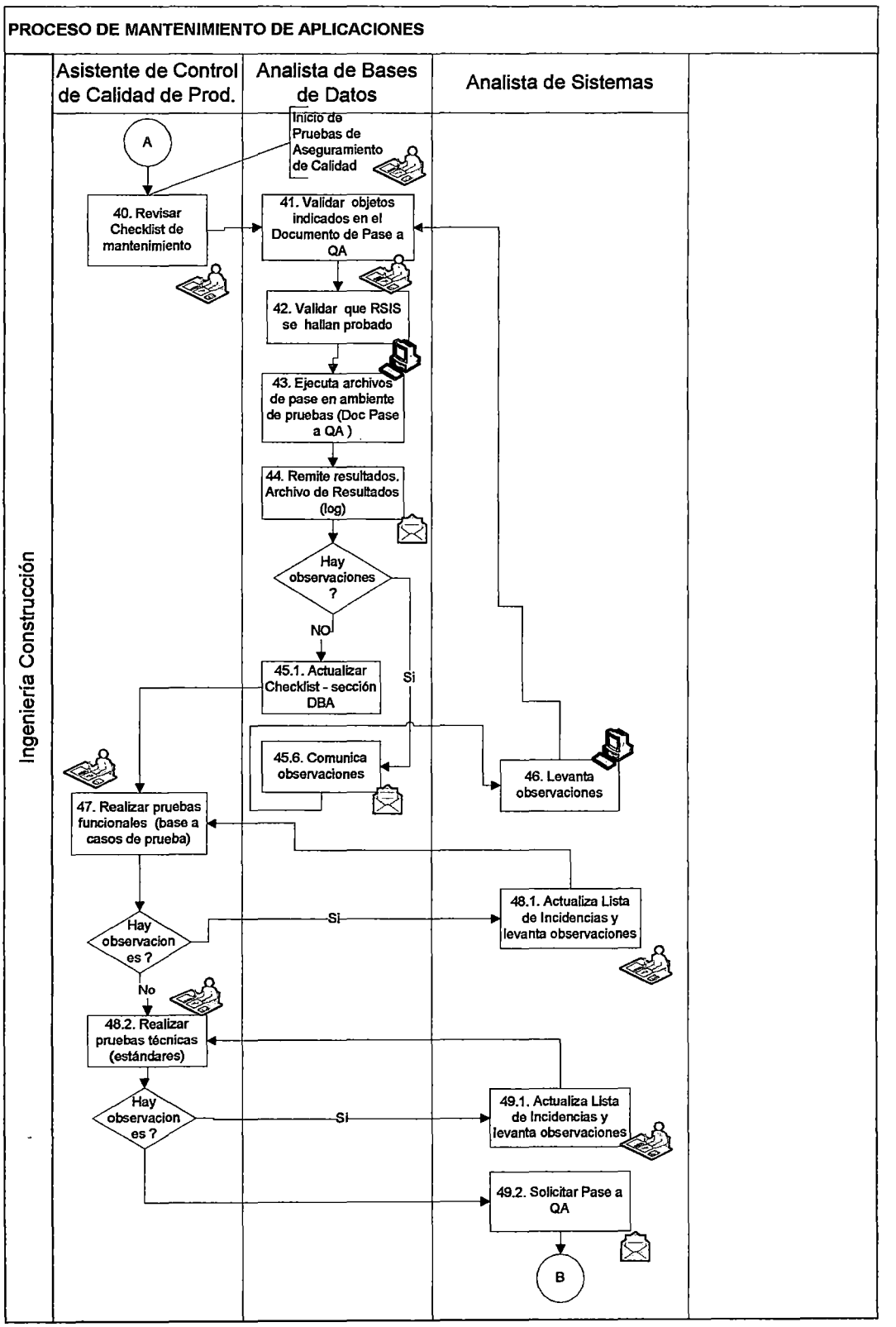
2. Diagrama del Proceso de Ingeniería de Mantenimiento

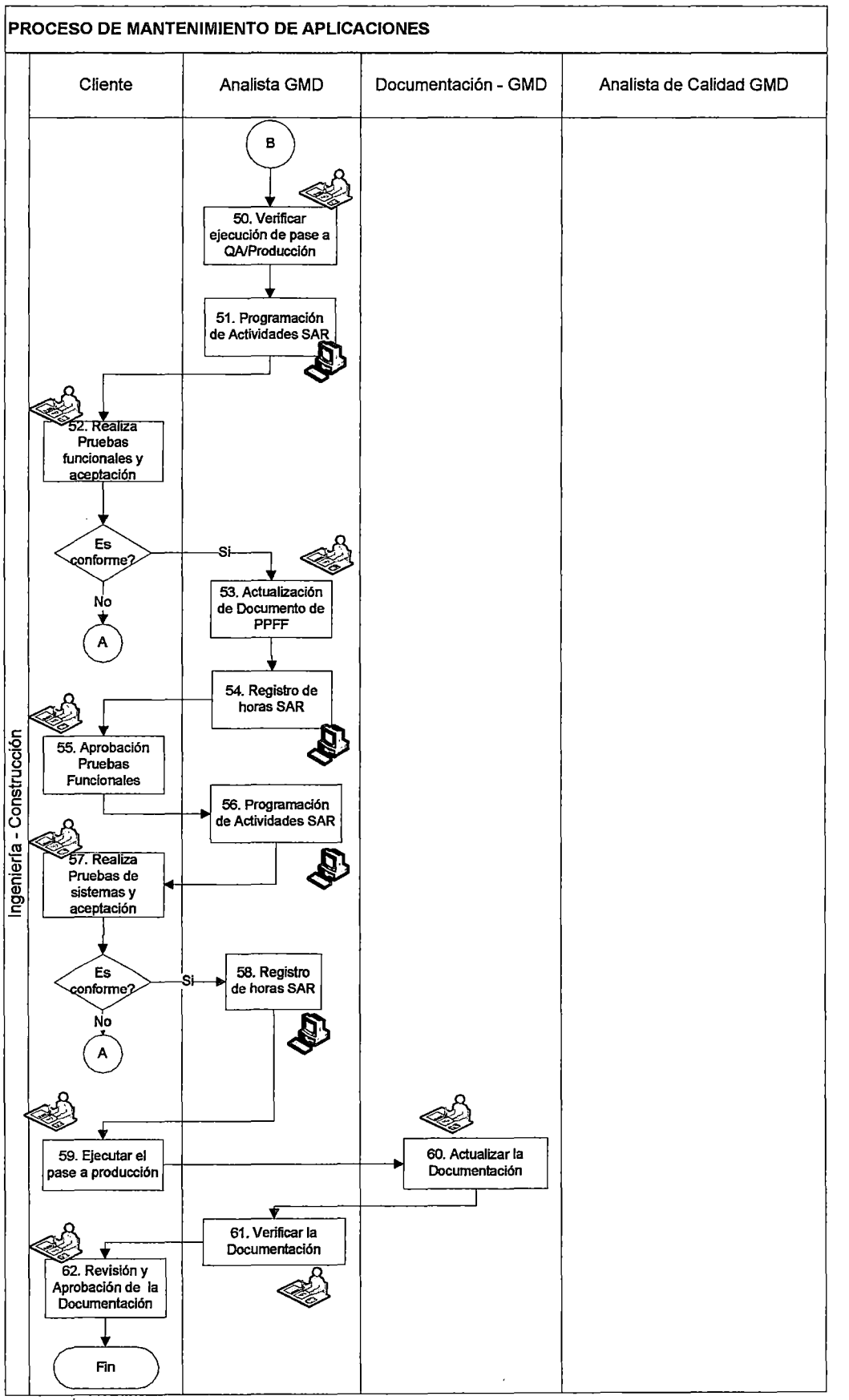
A. Fase de Incepción y Elaboración



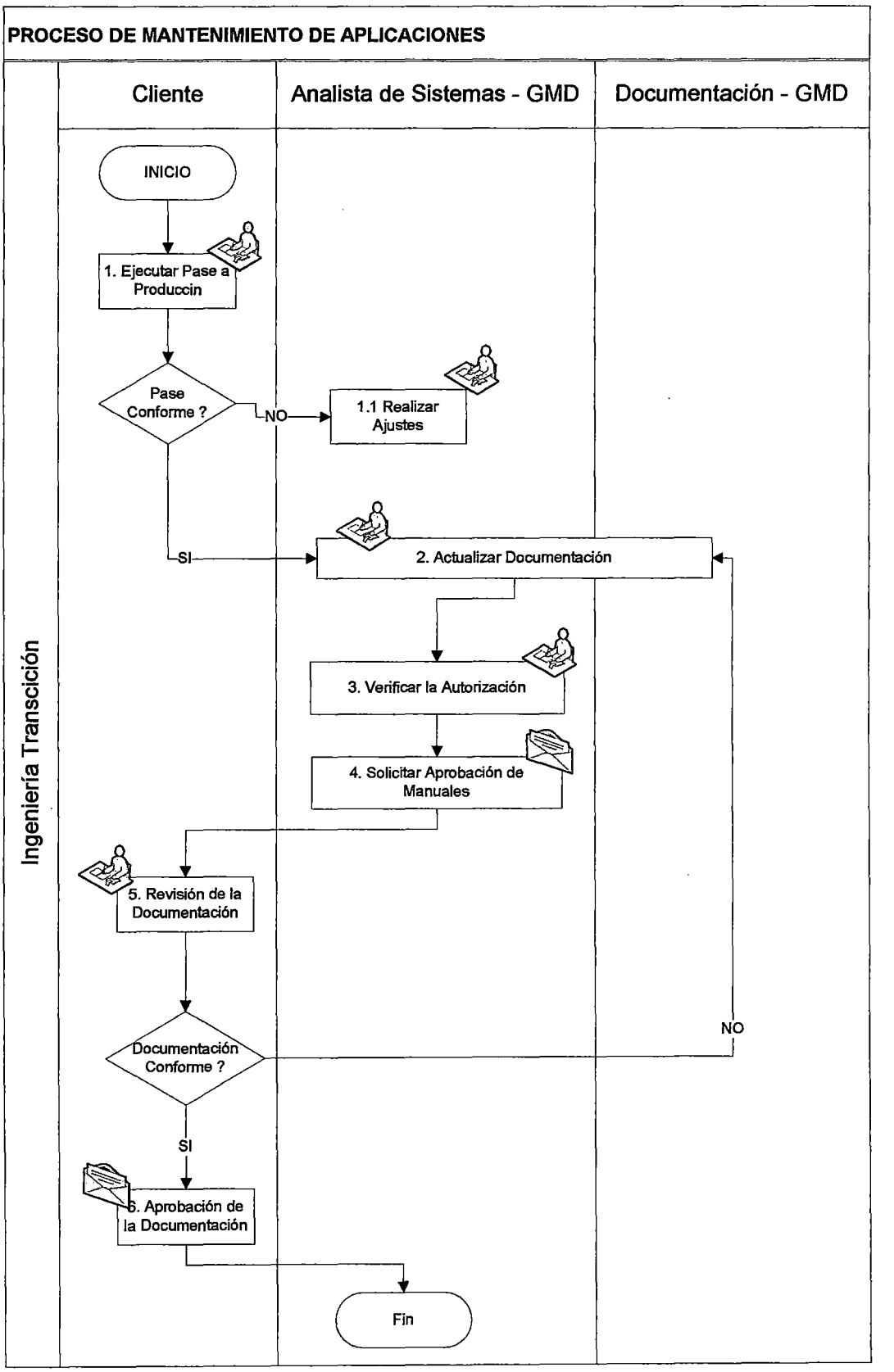
B. Fase de Construcción




PROCESO DE MANTENIMIENTO DE APLICACIONES
Ingeniería Construcción
Asistente de Control de Calidad de Prod.
Analista de Bases de Datos
Analista de Sistemas
A
Inicio de Pruebas de Aseguramiento de Calidad
40. Revisar Checklist de mantenimiento
41. Validar objetos indicados en el Documento de Pase a QA
42. Validar que RSIS se hallan probado
43. Ejecuta archivos de pase en ambiente de pruebas (Doc Pase a QA)
44. Remite resultados. Archivo de Resultados (log)
Hay observaciones ?
NO
Si
45.1. Actualizar Checklist - sección DBA
45.6. Comunica observaciones
46. Levanta observaciones
47. Realizar pruebas funcionales (base a casos de prueba)
Hay observacion es ?
Si
48.1. Actualiza Lista de Incidencias y levanta observaciones
No
48.2. Realizar pruebas técnicas (estándares)
Hay observacion es ?
Si
49.1. Actualiza Lista de Incidencias y levanta observaciones
49.2. Solicitar Pase a QA
B

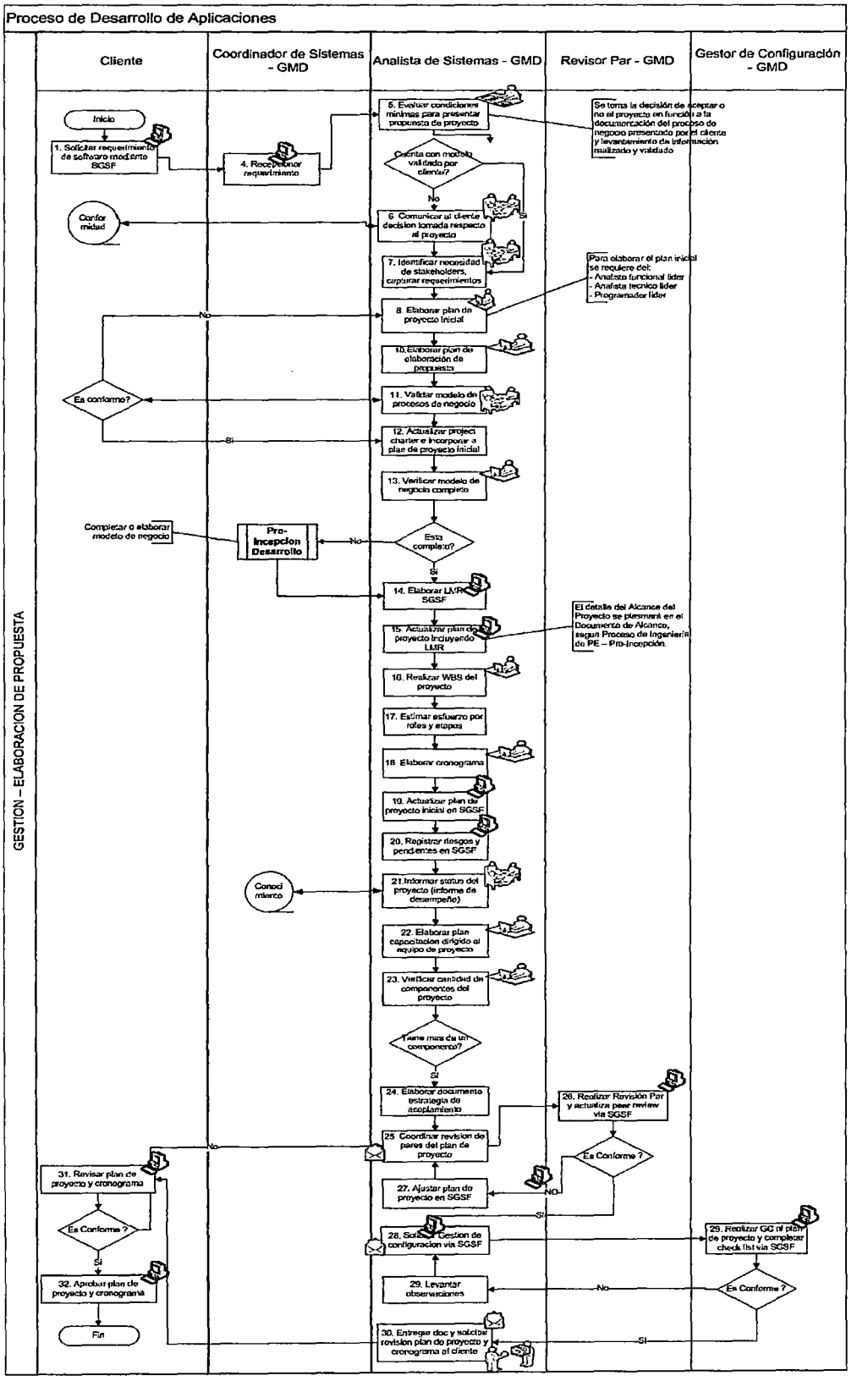

PROCESO DE MANTENIMIENTO DE APLICACIONES
Ingeniería - Construcción
Cliente
Analista GMD
Documentación - GMD
Analista de Calidad GMD
B
50. Verificar ejecución de pase a QA/Producción
51. Programación de Actividades SAR
52. Realiza Pruebas funcionales y aceptación
Es conforme?
Si
No
A
53. Actualización de Documento de PPFF
54. Registro de horas SAR
55. Aprobación Pruebas Funcionales
56. Programación de Actividades SAR
57. Realiza Pruebas de sistemas y aceptación
Es conforme?
Si
No
A
58. Registro de horas SAR
59. Ejecutar el pase a producción
60. Actualizar la Documentación
61. Verificar la Documentación
62. Revisión y Aprobación de la Documentación
Fin

Fase de Transición

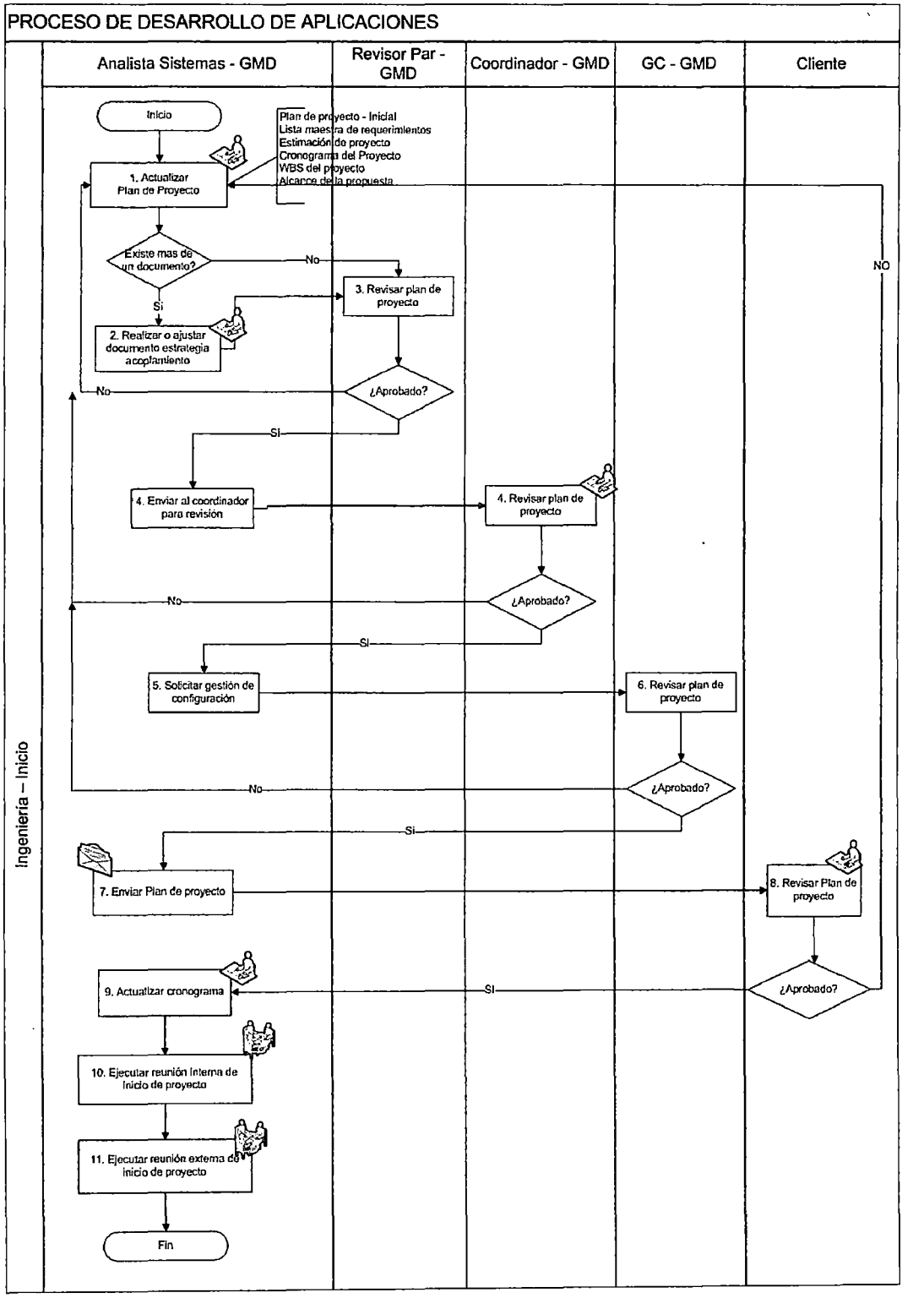


3. Diagrama del Proceso de Gestión de Desarrollo

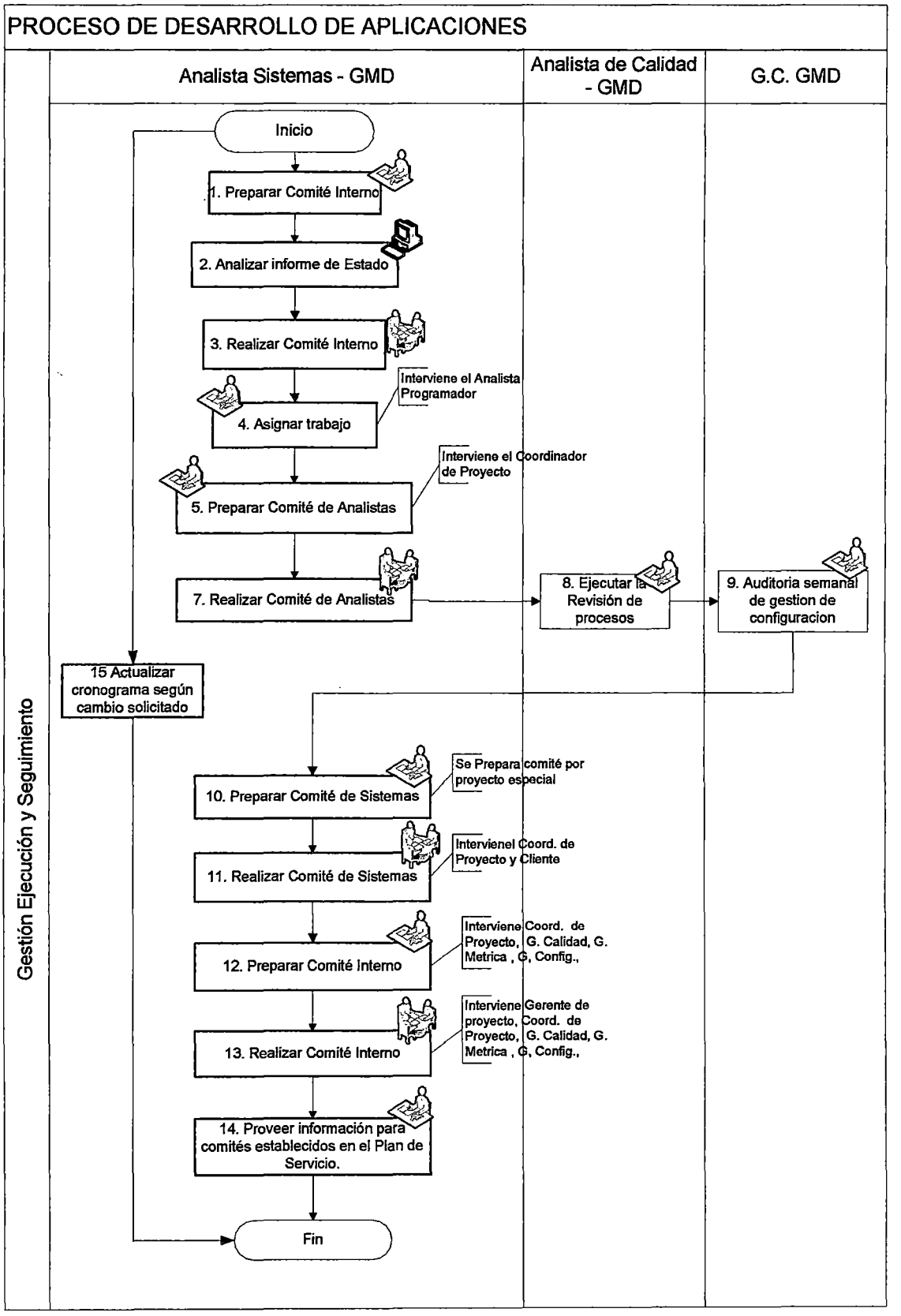
A. Fase de Elaboración de Propuesta e Incepción



B. Inicio


PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería – Inicio
Analista Sistemas - GMD
Revisor Par - GMD
Coordinador - GMD
GC - GMD
Cliente
Inicio
Plan de proyecto - Inicial
Lista maestra de requerimientos
Estimación de proyecto
Cronograma del Proyecto
WBS del proyecto
Alcance de la propuesta
1. Actualizar Plan de Proyecto
Existe mas de un documento?
No
Si
3. Revisar plan de proyecto
2. Realizar o ajustar documento estrategia acoplamiento
¿Aprobado?
No
Si
4. Enviar al coordinador para revisión
4. Revisar plan de proyecto
¿Aprobado?
No
Si
5. Solicitar gestión de configuración
6. Revisar plan de proyecto
¿Aprobado?
No
Si
7. Enviar Plan de proyecto
8. Revisar Plan de proyecto
¿Aprobado?
NO
Si
9. Actualizar cronograma
10. Ejecutar reunión interna de inicio de proyecto
11. Ejecutar reunión externa de inicio de proyecto
Fin

C. Fase de Ejecución y Seguimiento



D. Cierre

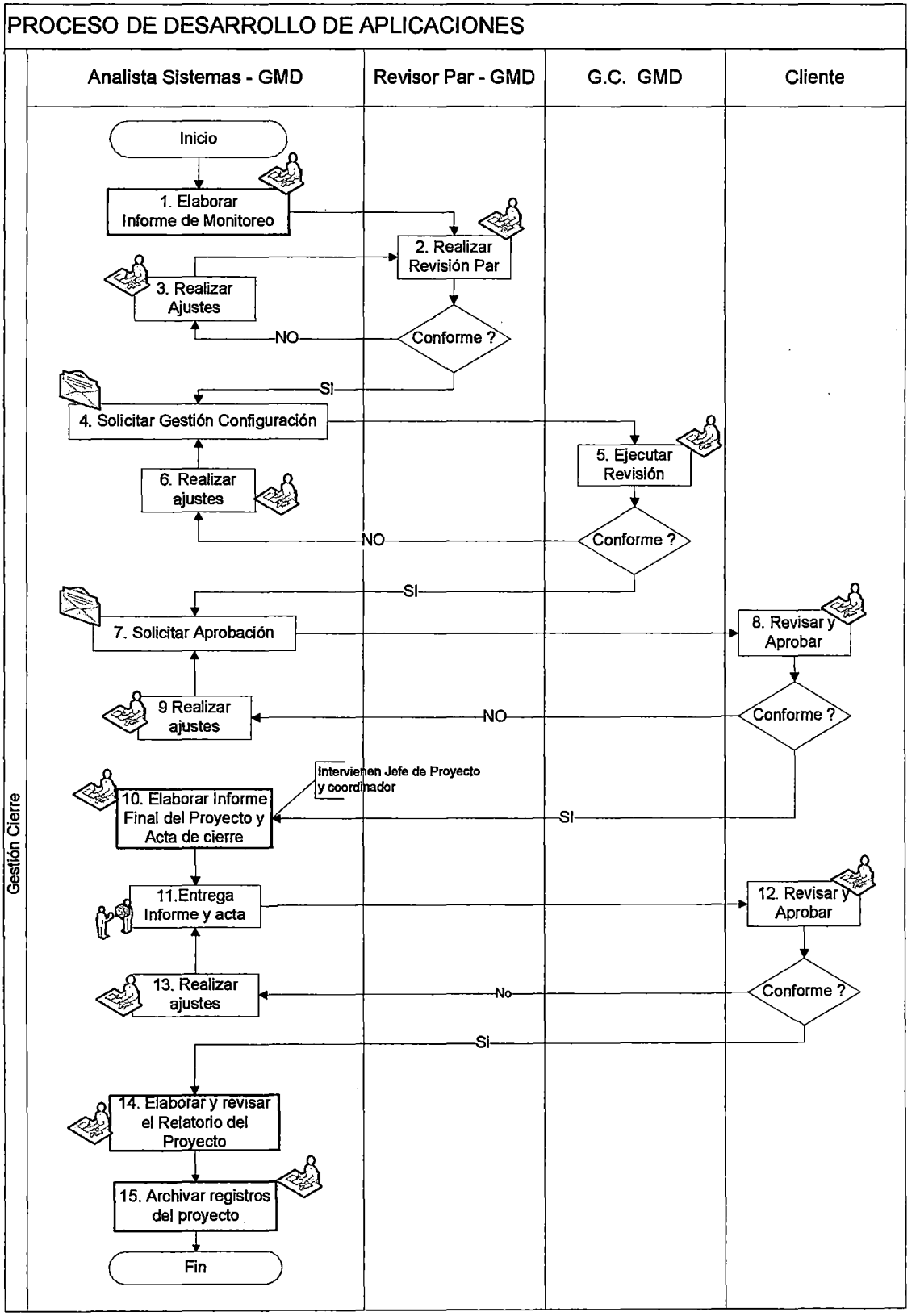
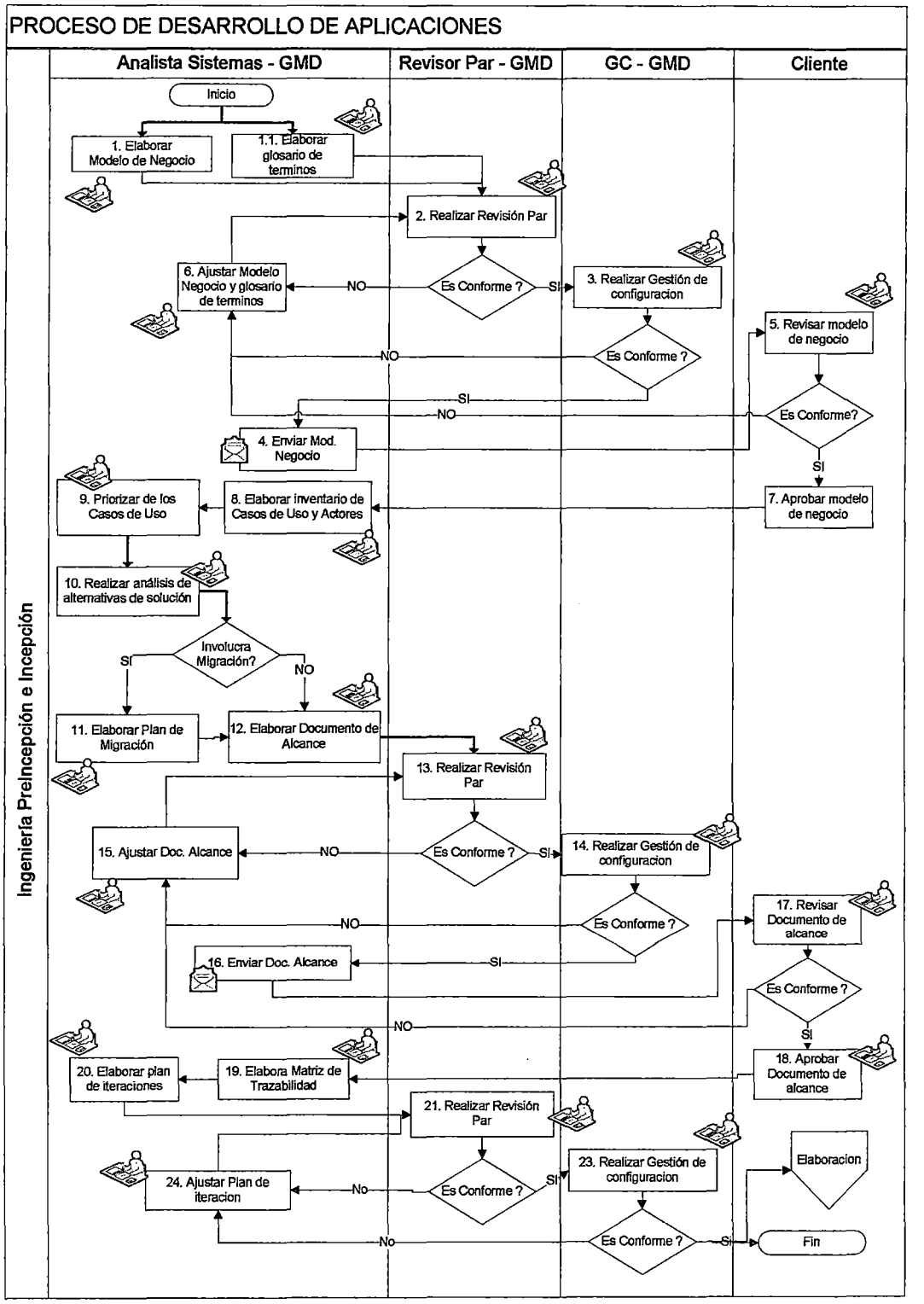
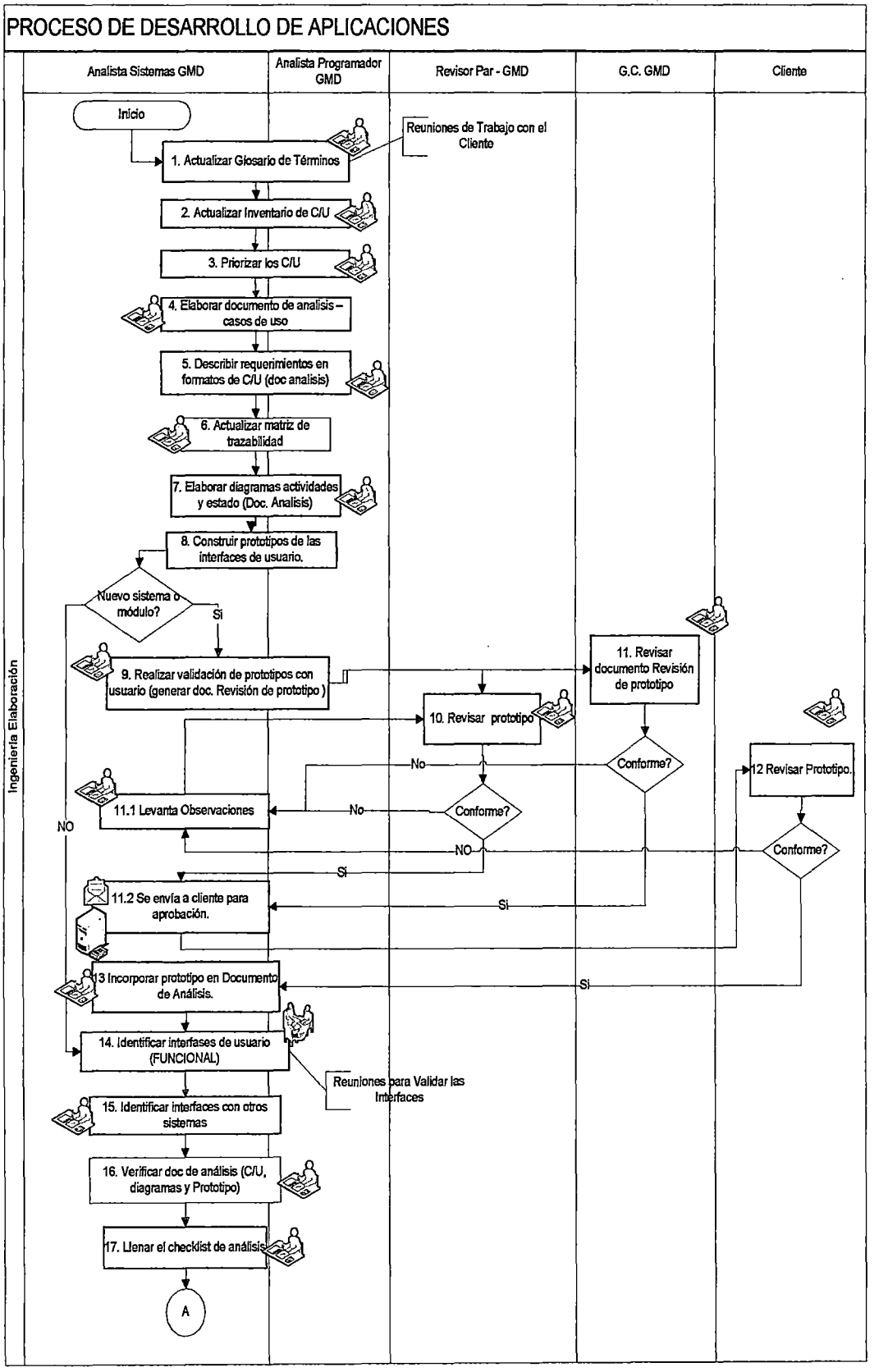

PROCESO DE DESARROLLO DE APLICACIONES
Analista Sistemas - GMD
Revisor Par - GMD
G.C. GMD
Cliente
Gestión Cierre
Inicio
1. Elaborar Informe de Monitoreo
2. Realizar Revisión Par
3. Realizar Ajustes
Conforme ?
NO
SI
4. Solicitar Gestión Configuración
5. Ejecutar Revisión
6. Realizar ajustes
Conforme ?
NO
SI
7. Solicitar Aprobación
8. Revisar y Aprobar
9 Realizar ajustes
Conforme ?
NO
Intervienen Jefe de Proyecto y coordinador
10. Elaborar Informe Final del Proyecto y Acta de cierre
SI
11.Entrega Informe y acta
12. Revisar y Aprobar
13. Realizar ajustes
Conforme ?
No
Si
14. Elaborar y revisar el Relatorio del Proyecto
15. Archivar registros del proyecto
Fin

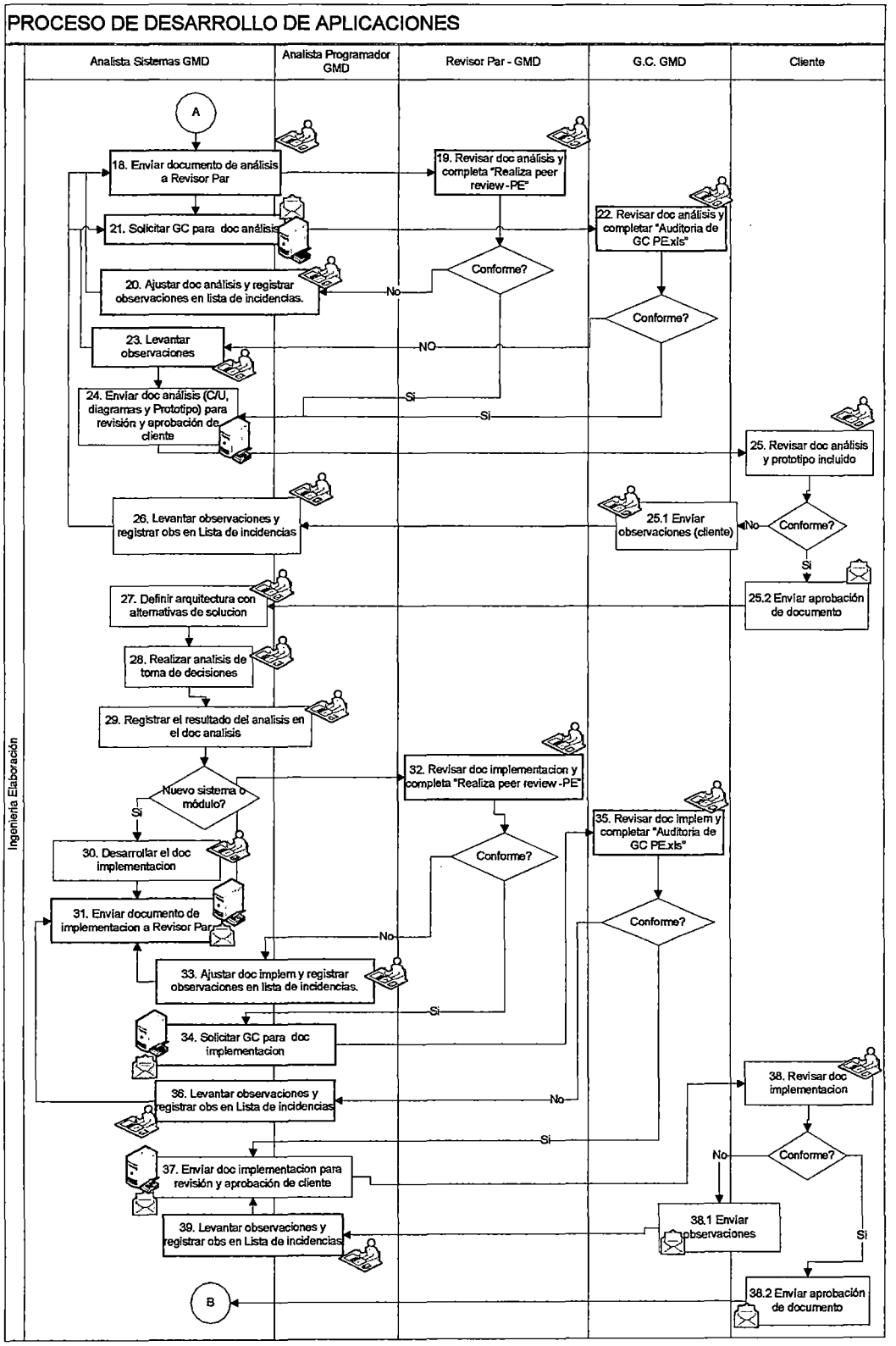
Diagrama del Proceso de Ingeniería de Desarrollo

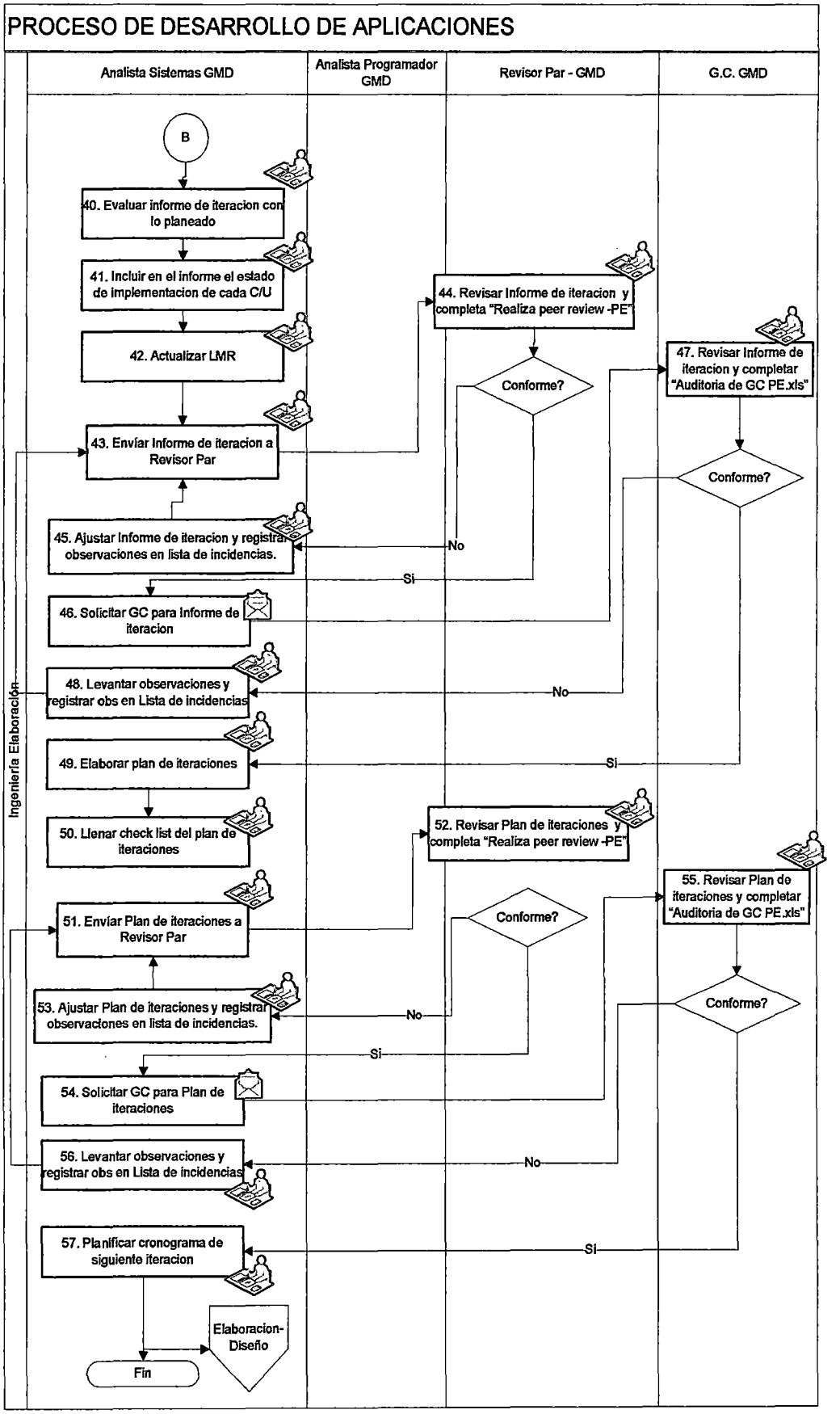
A. Fase de Pre-Incepción e Incepción

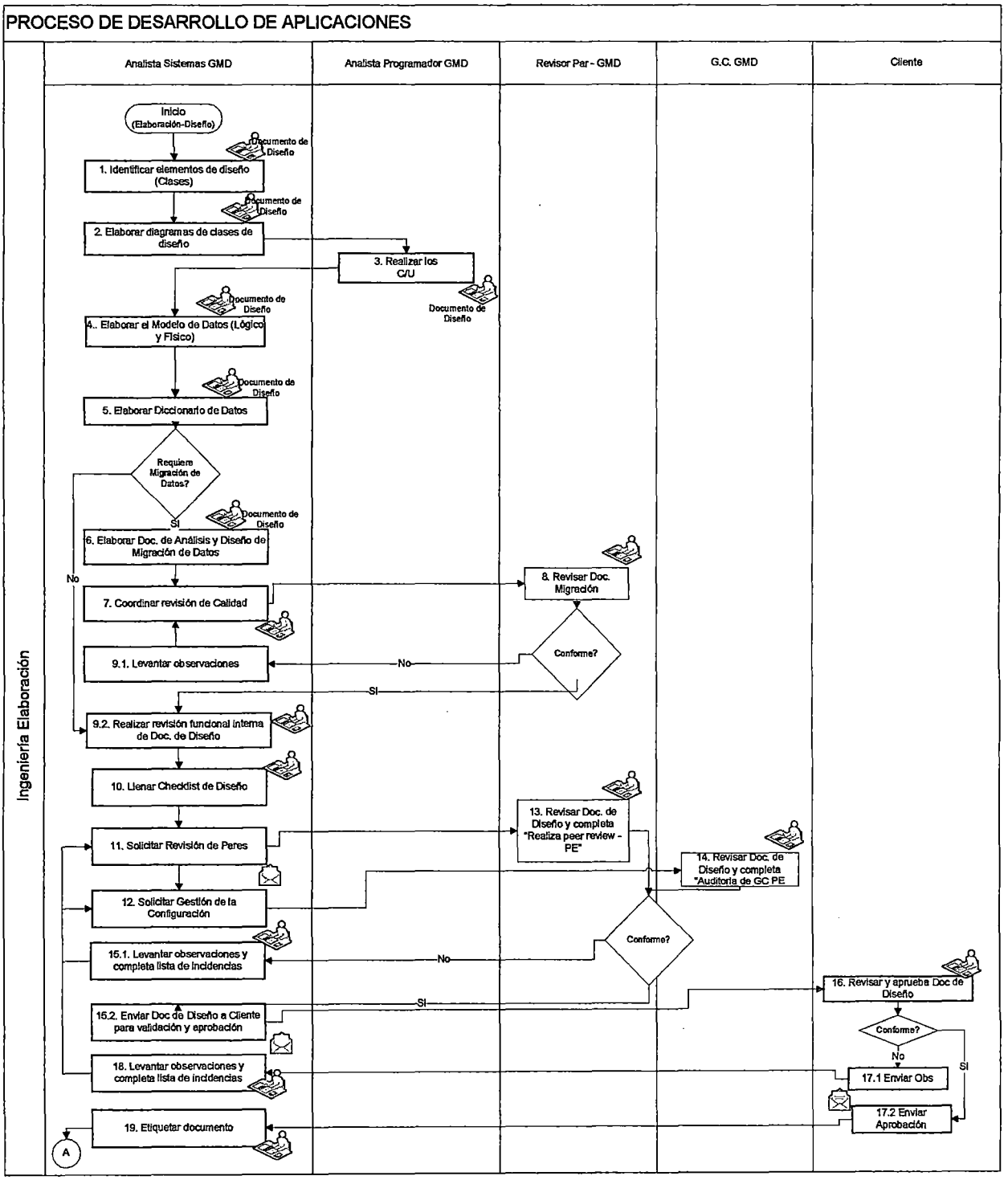


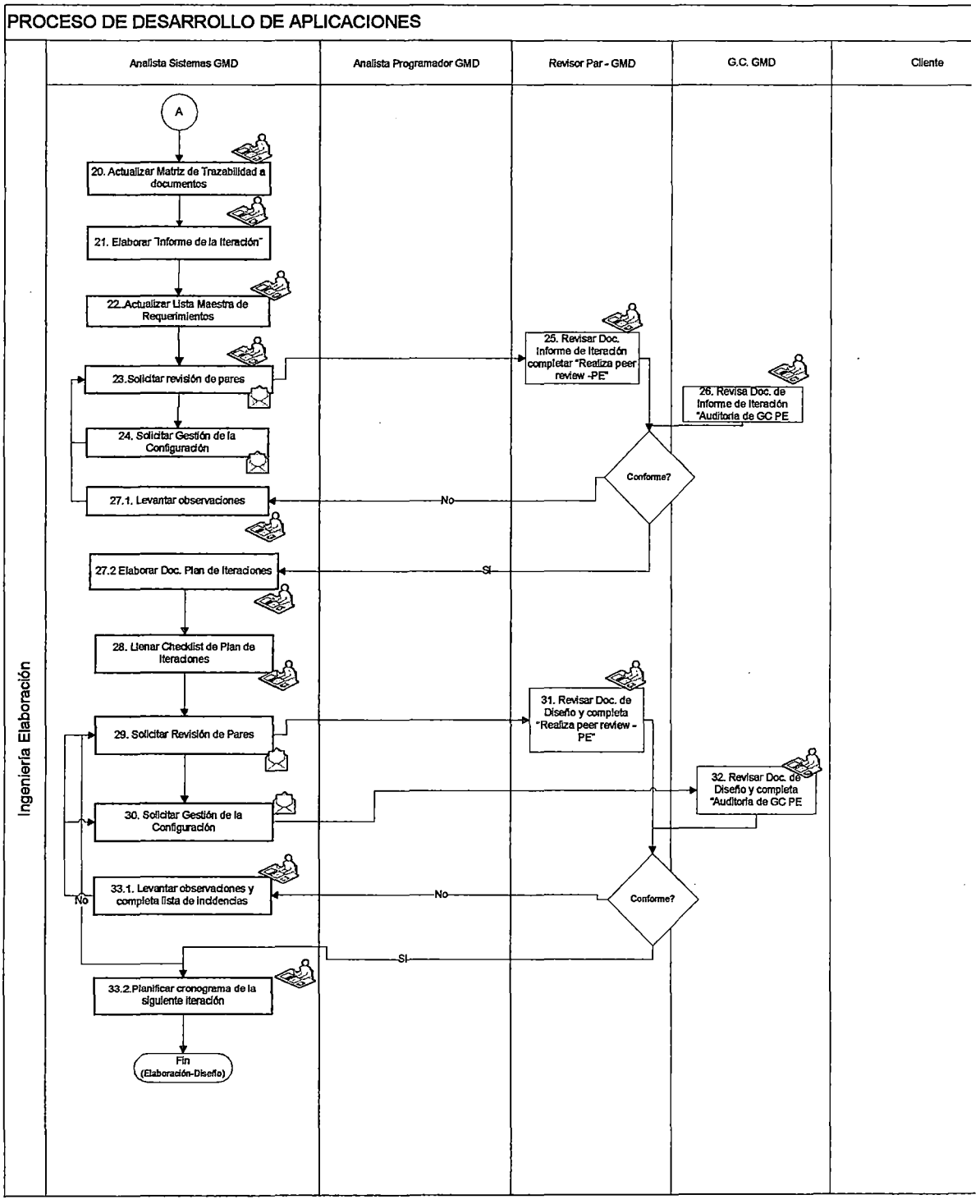
B. Fase de Elaboración



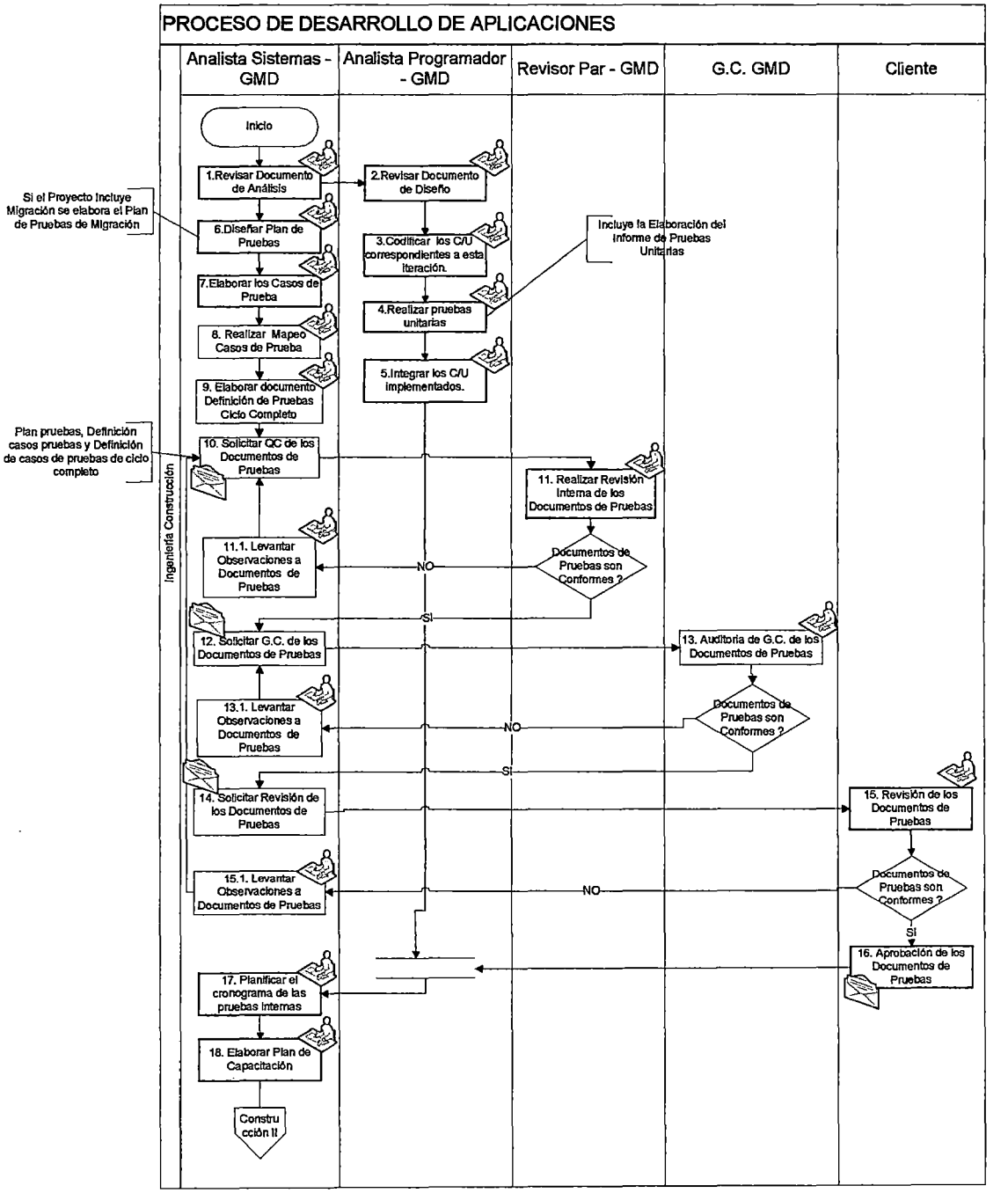

PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Elaboración
Analista Sistemas GMD
Analista Programador GMD
Revisor Par - GMD
G.C. GMD
Cliente
A
18. Enviar documento de análisis a Revisor Par
19. Revisar doc análisis y completa "Realiza peer review -PE"
21. Solicitar GC para doc análisis
22. Revisar doc análisis y completar "Auditoria de GC PE.xls"
Conforme?
20. Ajustar doc análisis y registrar observaciones en lista de incidencias.
No
Conforme?
23. Levantar observaciones
NO
24. Enviar doc análisis (C/U, diagramas y Prototipo) para revisión y aprobación de cliente
Si
Si
25. Revisar doc análisis y prototipo incluido
26. Levantar observaciones y registrar obs en Lista de incidencias
25.1 Enviar observaciones (cliente)
No
Conforme?
Si
25.2 Enviar aprobación de documento
27. Definir arquitectura con alternativas de solucion
28. Realizar analisis de toma de decisiones
29. Registrar el resultado del analisis en el doc analisis
32. Revisar doc implementacion y completa "Realiza peer review -PE"
Nuevo sistema o módulo?
Si
35. Revisar doc implem y completar "Auditoria de GC PE.xls"
Conforme?
30. Desarrollar el doc implementacion
Conforme?
31. Enviar documento de implementacion a Revisor Par
No
33. Ajustar doc implem y registrar observaciones en lista de incidencias.
Si
34. Solicitar GC para doc implementacion
38. Revisar doc implementacion
36. Levantar observaciones y registrar obs en Lista de incidencias
No
Si
Conforme?
No
37. Enviar doc implementacion para revisión y aprobación de cliente
38.1 Enviar observaciones
Si
39. Levantar observaciones y registrar obs en Lista de incidencias
38.2 Enviar aprobación de documento
B

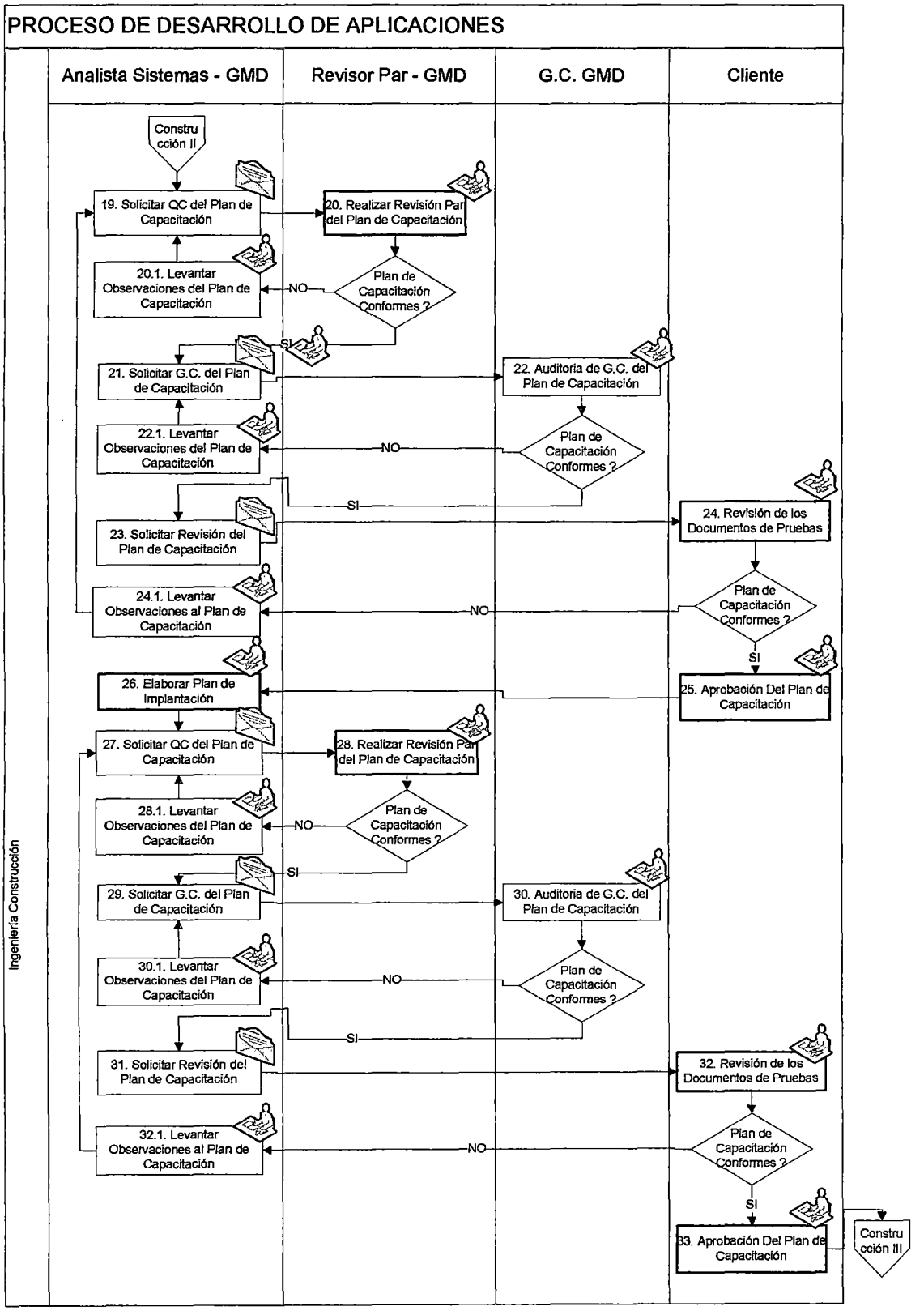

PROCESO DE DESARROLLO DE APLICACIONES
Analista Sistemas GMD
Analista Programador GMD
Revisor Par - GMD
G.C. GMD
Ingeniería Elaboración
B
40. Evaluar informe de iteracion con lo planeado
41. Incluir en el informe el estado de implementacion de cada C/U
42. Actualizar LMR
43. Envíar Informe de iteracion a Revisor Par
44. Revisar Informe de iteracion y completa "Realiza peer review -PE"
Conforme?
47. Revisar Informe de iteracion y completar "Auditoria de GC PE.xls"
Conforme?
45. Ajustar Informe de iteracion y registrar observaciones en lista de incidencias.
No
Si
46. Solicitar GC para Informe de iteracion
48. Levantar observaciones y registrar obs en Lista de incidencias
No
49. Elaborar plan de iteraciones
Si
50. Llenar check list del plan de iteraciones
52. Revisar Plan de iteraciones y completa "Realiza peer review -PE"
55. Revisar Plan de iteraciones y completar "Auditoria de GC PE.xls"
51. Envíar Plan de iteraciones a Revisor Par
Conforme?
Conforme?
53. Ajustar Plan de iteraciones y registrar observaciones en lista de incidencias.
No
Si
54. Solicitar GC para Plan de iteraciones
56. Levantar observaciones y registrar obs en Lista de incidencias
No
57. Planificar cronograma de siguiente iteracion
Si
Elaboracion-Diseño
Fin

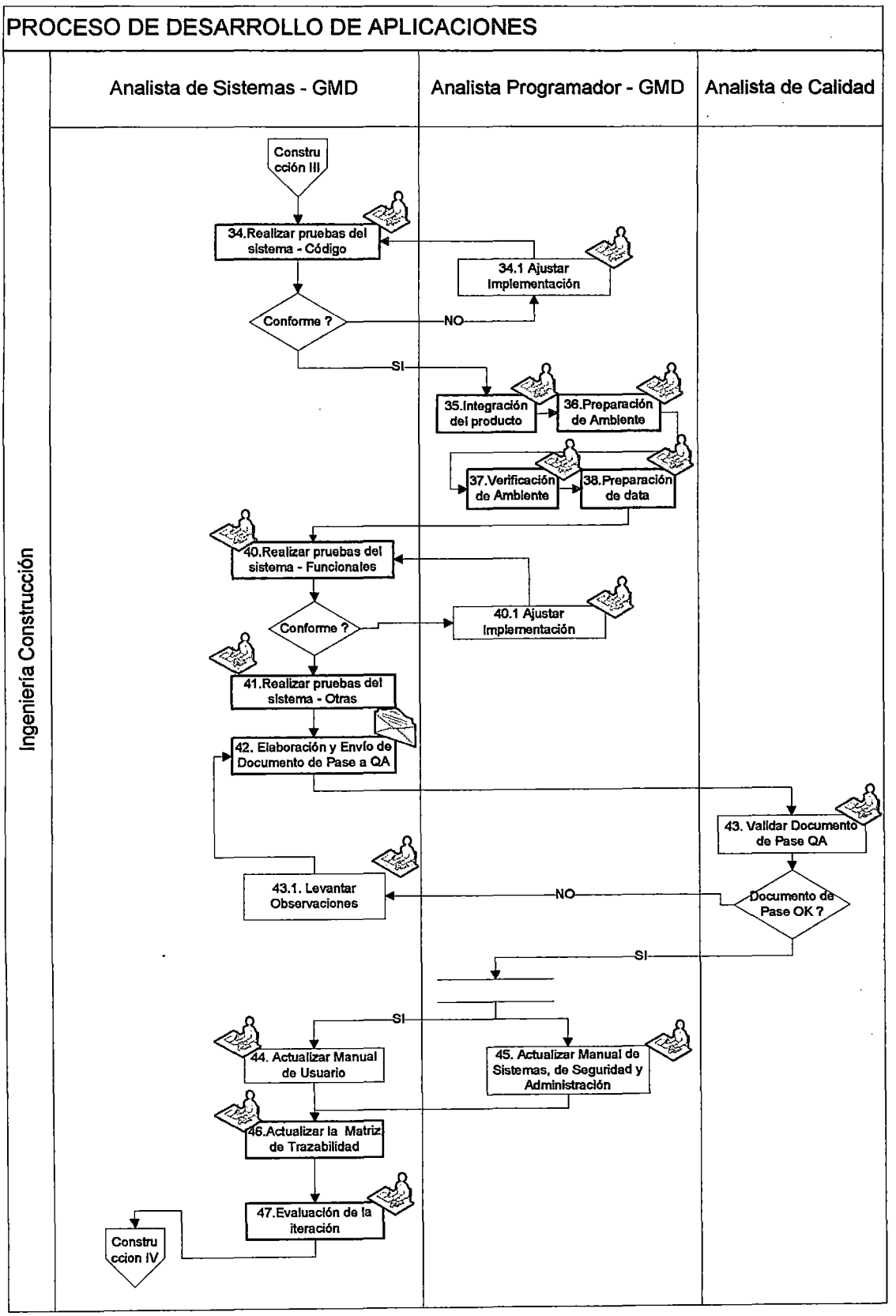

PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Elaboración
Analista Sistemas GMD
Analista Programador GMD
Revisor Par - GMD
G.C. GMD
Cliente
Inicio (Elaboración-Diseño)
Documento de Diseño
1. Identificar elementos de diseño (Clases)
2. Elaborar diagramas de clases de diseño
3. Realizar los C/U
4.. Elaborar el Modelo de Datos (Lógico y Físico)
5. Elaborar Diccionario de Datos
Requiere Migración de Datos?
SI
No
6. Elaborar Doc. de Análisis y Diseño de Migración de Datos
7. Coordinar revisión de Calidad
8. Revisar Doc. Migración
Conforme?
No
SI
9.1. Levantar observaciones
9.2. Realizar revisión funcional interna de Doc. de Diseño
10. Llenar Checklist de Diseño
11. Solicitar Revisión de Peres
13. Revisar Doc. de Diseño y completa "Realiza peer review - PE"
14. Revisar Doc. de Diseño y completa "Auditoria de GC PE
12. Solicitar Gestión de la Configuración
Conforme?
No
15.1. Levantar observaciones y completa lista de incidencias
SI
15.2. Enviar Doc de Diseño a Cliente para validación y aprobación
16. Revisar y aprueba Doc de Diseño
Conforme?
No
SI
17.1 Enviar Obs
17.2 Enviar Aprobación
18. Levantar observaciones y completa lista de incidencias
19. Etiquetar documento
A

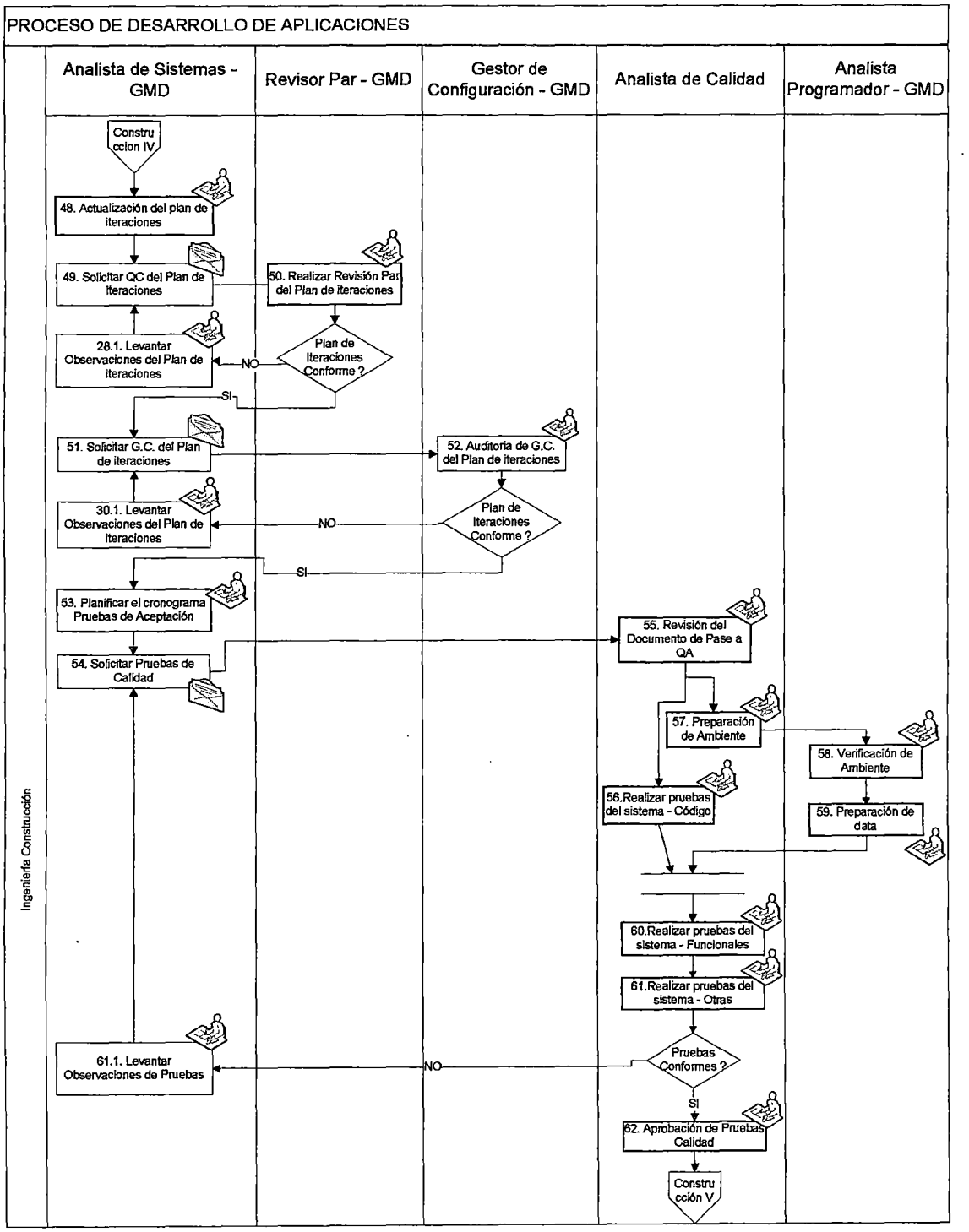

PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Elaboración
Analista Sistemas GMD
Analista Programador GMD
Revisor Par - GMD
G.C. GMD
Cliente
A
20. Actualizar Matriz de Trazabilidad a documentos
21. Elaborar "Informe de la Iteración"
22. Actualizar Lista Maestra de Requerimientos
23. Solicitar revisión de pares
24. Solicitar Gestión de la Configuración
25. Revisar Doc. Informe de Iteración completar "Realiza peer review -PE"
26. Revisa Doc. de Informe de Iteración "Auditoria de GC PE
Conforme?
No
27.1. Levantar observaciones
Si
27.2 Elaborar Doc. Plan de Iteraciones
28. Llenar Checklist de Plan de Iteraciones
29. Solicitar Revisión de Pares
30. Solicitar Gestión de la Configuración
31. Revisar Doc. de Diseño y completa "Realiza peer review - PE"
32. Revisar Doc. de Diseño y completa "Auditoria de GC PE
Conforme?
No
33.1. Levantar observaciones y completa lista de incidencias
Si
33.2.Planificar cronograma de la siguiente iteración
Fin (Elaboración-Diseño)

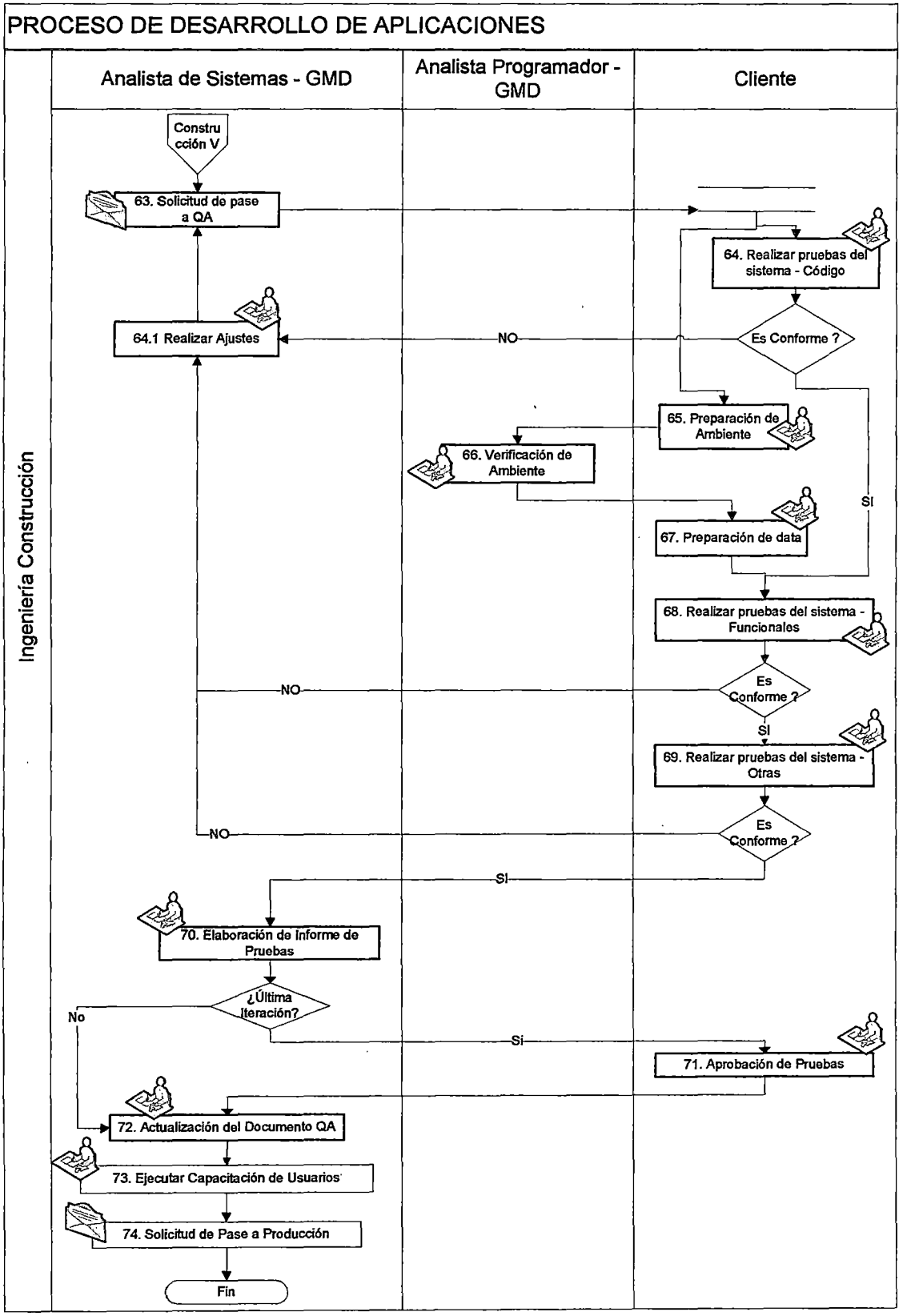
C. Fase de Construcción



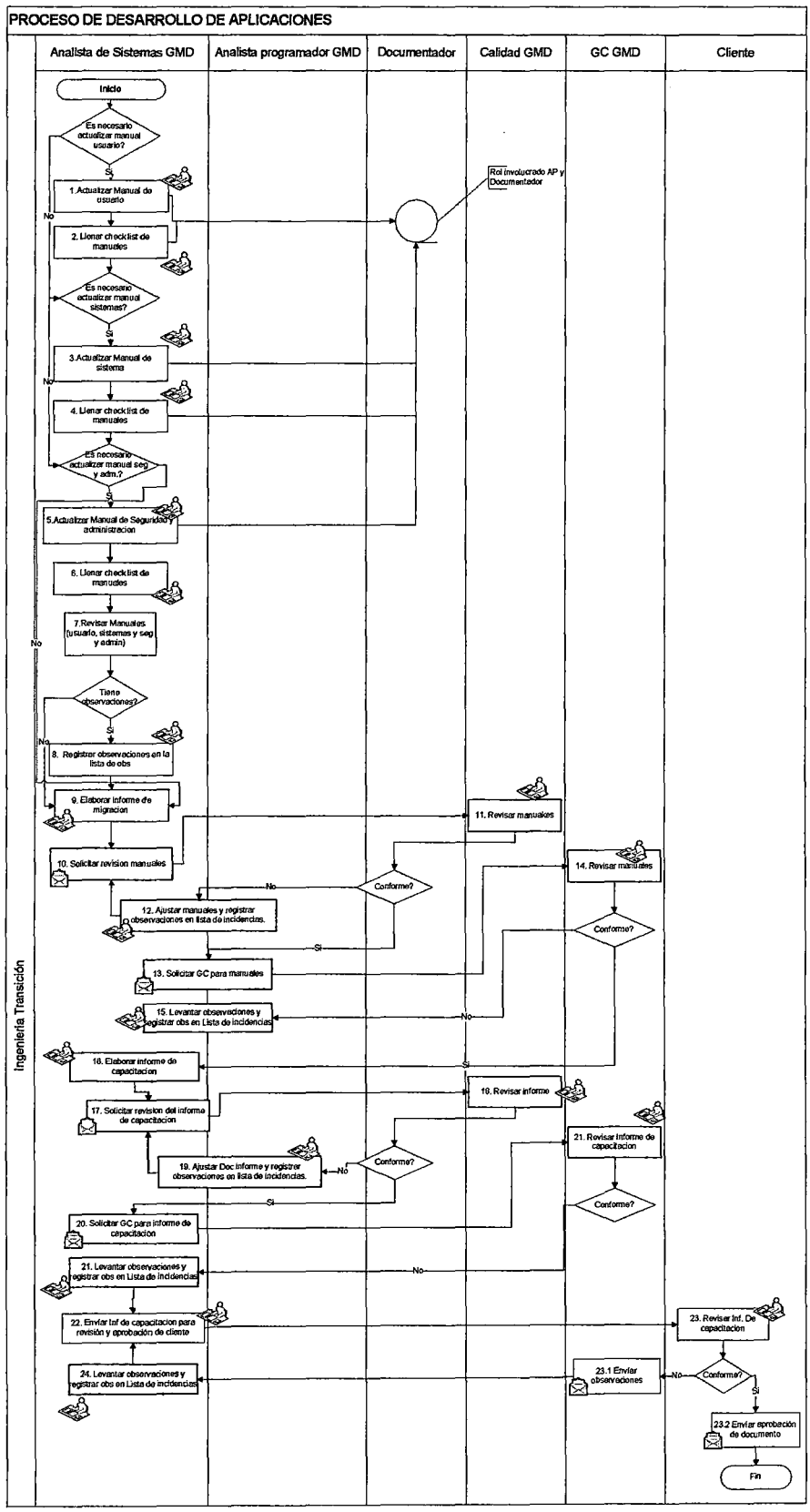

PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Construcción
Analista Sistemas - GMD
Revisor Par - GMD
G.C. GMD
Cliente
Construcción II
19. Solicitar QC del Plan de Capacitación
20. Realizar Revisión Par del Plan de Capacitación
20.1. Levantar Observaciones del Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
21. Solicitar G.C. del Plan de Capacitación
22. Auditoria de G.C. del Plan de Capacitación
22.1. Levantar Observaciones del Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
23. Solicitar Revisión del Plan de Capacitación
24. Revisión de los Documentos de Pruebas
24.1. Levantar Observaciones al Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
26. Elaborar Plan de Implantación
25. Aprobación Del Plan de Capacitación
27. Solicitar QC del Plan de Capacitación
28. Realizar Revisión Par del Plan de Capacitación
28.1. Levantar Observaciones del Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
29. Solicitar G.C. del Plan de Capacitación
30. Auditoria de G.C. del Plan de Capacitación
30.1. Levantar Observaciones del Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
31. Solicitar Revisión del Plan de Capacitación
32. Revisión de los Documentos de Pruebas
32.1. Levantar Observaciones al Plan de Capacitación
NO
Plan de Capacitación Conformes ?
SI
33. Aprobación Del Plan de Capacitación
Construcción III


PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Construcción
Analista de Sistemas - GMD
Analista Programador - GMD
Analista de Calidad
Construcción III
34.Realizar pruebas del sistema - Código
34.1 Ajustar Implementación
Conforme ?
NO
SI
35.Integración del producto
36.Preparación de Ambiente
37.Verificación de Ambiente
38.Preparación de data
40.Realizar pruebas del sistema - Funcionales
Conforme ?
40.1 Ajustar Implementación
41.Realizar pruebas del sistema - Otras
42. Elaboración y Envío de Documento de Pase a QA
43. Validar Documento de Pase QA
43.1. Levantar Observaciones
NO
Documento de Pase OK ?
SI
SI
44. Actualizar Manual de Usuario
45. Actualizar Manual de Sistemas, de Seguridad y Administración
46.Actualizar la Matriz de Trazabilidad
47.Evaluación de la iteración
Construccion IV


PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Construcción
Analista de Sistemas - GMD
Revisor Par - GMD
Gestor de Configuración - GMD
Analista de Calidad
Analista Programador - GMD
Construcion IV
48. Actualización del plan de iteraciones
49. Solicitar QC del Plan de Iteraciones
50. Realizar Revisión Par del Plan de iteraciones
Plan de Iteraciones Conforme ?
NO
28.1. Levantar Observaciones del Plan de iteraciones
SI
51. Solicitar G.C. del Plan de iteraciones
52. Auditoria de G.C. del Plan de iteraciones
Plan de Iteraciones Conforme ?
NO
30.1. Levantar Observaciones del Plan de iteraciones
SI
53. Planificar el cronograma Pruebas de Aceptación
54. Solicitar Pruebas de Calidad
55. Revisión del Documento de Pase a QA
57. Preparación de Ambiente
58. Verificación de Ambiente
59. Preparación de data
56.Realizar pruebas del sistema - Código
60.Realizar pruebas del sistema - Funcionales
61.Realizar pruebas del sistema - Otras
Pruebas Conformes ?
NO
61.1. Levantar Observaciones de Pruebas
SI
62. Aprobación de Pruebas Calidad
Construcción V


PROCESO DE DESARROLLO DE APLICACIONES
Ingeniería Construcción
Analista de Sistemas - GMD
Analista Programador - GMD
Cliente
Construcción V
63. Solicitud de pase a QA
64. Realizar pruebas del sistema - Código
Es Conforme ?
NO
64.1 Realizar Ajustes
SI
65. Preparación de Ambiente
66. Verificación de Ambiente
67. Preparación de data
68. Realizar pruebas del sistema - Funcionales
Es Conforme ?
NO
SI
69. Realizar pruebas del sistema - Otras
Es Conforme ?
NO
SI
70. Elaboración de Informe de Pruebas
¿Última Iteración?
No
Si
71. Aprobación de Pruebas
72. Actualización del Documento QA
73. Ejecutar Capacitación de Usuarios
74. Solicitud de Pase a Producción
Fin

D. Fase de Transición



ANEXO 3

RESULTADOS DE RETROSPECTIVA

ITERACIÓN 1

	+5	-10	-7	+3	0	7	-5	-7	-5	+10	-5	+7
Historia de Usuario	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	**Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior**	**Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar**	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	**Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan**	**Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN**	**Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.**	**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	**Cada error cometido**	**Cada error corregido**	**Cada error no corregido**	**Cada error corregido pero que se deba mejorar**
Adicionar los campos Proceso y Ejecución a la tabla de detalle fraccionado.	Realización de los scripts	Actualización de versiones	Estándares de programación		estándares de programación	generación de queries de reversión	falta de coordinación interna	generación de queries de reversión			cambio de etiqueta	
Registro del proceso y ejecución por el proceso de carga de cuenta	Realización de los scripts	Actualización de versiones										Actualización del saldo para el casos de

Historia de Usuario		corriente.
Los aspectos que se han realizado correctamente en cada ITERACIÓN	+5	
Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10	
Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7	
Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	+3	
Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0	
Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7	
Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5	
Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7	
Cada error cometido	-5	
Cada error corregido	+10	
Cada error no corregido	-5	
Cada error corregido pero que se deba mejorar	+7	la Planilla nuevo cálculo

Puntaje TOTAL
10
-20
-7
0
0
0
0
-7
0
0
-5
7
-22

ITERACIÓN 2

Historia de Usuario	+5 Los aspectos que se han realizado correctamente en cada ITERACIÓN	-10 Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-7 Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	+3 Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	0 Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	7 Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	-5 Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-7 Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-5 Cada error cometido	+10 Cada error corregido	-5 Cada error no corregido	+7 Cada error corregido pero que se deba mejorar
Registro del proceso y ejecución mediante la aprobación del motivo de solicitud cambio de tutor.	Codificación				Refactorización			Ejecución de simulación de pase	No se consideró el código de proceso			

Registro del proceso y ejecución mediante la aprobación del motivo de solicitud modificación y anulación de saldos.	Registro del proceso y ejecución mediante la aprobación del motivo de solicitud de cambio de riesgo.	Historia de Usuario	
Codificación	Codificación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
Actualización de versiones		**Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior**	-10
	Estándares de programación	**Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar**	-7
Programación en pareja y interacción con el cliente e integración continua		**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
		Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
		Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
Falta de coordinación interna		**Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.**	-5
		Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7
		Cada error cometido	-5
		Cada error corregido	+10
Colocación de la historia de modificaciones en los programas		**Cada error no corregido**	-5
		Cada error corregido pero que se deba mejorar	+7

Historia de Usuario	
Los aspectos que se han realizado correctamente en cada ITERACIÓN	+5
Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	+3
Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7
Cada error cometido	-5
Cada error corregido	+10
Cada error no corregido	-5
Cada error corregido pero que se deba mejorar	+7

Registro del proceso y ejecución mediante la aprobación del motivo de regularización de pensiones.	Registro del proceso y ejecución mediante la aprobación del motivo de unificación de cuentas.	Puntaje
Codificación	Codificación	25
Pruebas Internas		-20
		-7
	Programación en pareja y interacción con el cliente e integración continua.	6
Refactorización		0
Generación de queries de reversión		7
Generación de queries de reversión		-10
Generación de queries de reversión	Ejecución de simulación de pase	-21
		-5
		0
		-5
		0

TOTAL
-30

ITERACIÓN 3

Historia de Usuario	
Actualización del saldo en la historia fraccionada para el caso de una modificación de saldo a un monto menor.	
Los aspectos que se han realizado correctamente en cada ITERACIÓN	+5
Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	+3
Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7
Cada error cometido	-5
Cada error corregido	+10
Cada error no corregido	-5
Cada error corregido pero que se deba mejorar	+7

Realización de los scripts
Estimación de tiempos
Cruce de versiones

Actualización de versiones

Programación en pareja y interacción con el cliente e integración continua

Estándares de programación

Generación de queries de reversión

Falta de coordinación interna.

Generación de queries de reversión.

No anula correctamente el detalle de historia fraccionada.

	+5	-10	-7	+3	0	7	-5	-7	-5	+10	-5	+7
Historia de Usuario	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	**Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior**	**Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar**	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	**Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan**	**Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN**	**Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.**	**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	**Cada error cometido**	**Cada error corregido**	**Cada error no corregido**	**Cada error corregido pero que se deba mejorar**
Revertir la historia de saldos y fraccionados tomando en cuenta que si existiere un saldo que fue anulado por otro proceso, se debe emitir un mensaje que la reversión se tiene que hacer en forma manual.	Realización de los scripts Estimación de tiempos Cruce de versiones		Estándares de programación	Programación en pareja y interacción con el cliente e integración continua								Actualización del saldo para el casos de la Planilla nuevo cálculo
Puntaje	**10**	**-10**	**-7**	**6**	**0**	**7**	**-5**	**-7**	**-5**	**0**	**0**	**7**
TOTAL	**-4**											

ITERACIÓN 4

Anulación de la Orden de Pago.	Modificación del Filtro.	Historia de Usuario	
Codificación	Codificación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
		Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
Estándares de programación		**Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar**	-7
	Programación en pareja y interacción con el cliente e integración continua	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
	Refactorización	**Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan**	0
		Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
		Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
	Ejecución de simulación de pase	**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	-7
No anula las planillas adicionales		**Cada error cometido**	-5
Se corrigió la anulación de planillas adicionales		**Cada error corregido**	+10
		Cada error no corregido	-5
		Cada error corregido pero que se deba mejorar	+7

Revertir Saldo y Cuota de Devengado Planilla Adicionales.	Revertir Saldo y Cuota de Devengado Planilla Normal incluye la ley 27803.	**Historia de Usuario**	
Estimación	Estimación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
		Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
		Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
	Programación en pareja y interacción con el cliente e integración continua	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
Refactorización		**Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan**	0
Generación de queries de reversión		**Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN**	7
		Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
Generación de queries de reversión		**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	-7
		Cada error cometido	-5
		Cada error corregido	+10
	Colocación de la historia de modificaciones en los programas	**Cada error no corregido**	-5
		Cada error corregido pero que se deba mejorar	+7

Anulación de Consignaciones por medio de la opción de depuración de fallecidos y anulación de órdenes de pago puntual y masivo.	Historia de Usuario	
Codificación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
	Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
	Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
Programación en pareja y interacción con el cliente e integración continua	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
	Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
	Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
	Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
Ejecución de simulación de pase	**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	-7
	Cada error cometido	-5
	Cada error corregido	+10
	Cada error no corregido	-5
	Cada error corregido pero que se deba mejorar	+7

Anulación de consignaciones mediante la opción del módulo de consignaciones	Historia de Usuario	
Puntaje TOTAL		
Script de reversión 25 11	Los aspectos que se han realizado correctamente en cada ITERACIÓN	+5
0	Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
-7	Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
12 Programación en pareja y interacción con el cliente e integración continua	Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	+3
0	Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
7	Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7
0	Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
-21	Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7
-5	Cada error cometido	-5
10	Cada error corregido	+10
-10 No actualiza la reversión de saldos	Cada error no corregido	-5
0	Cada error corregido pero que se deba mejorar	+7

ITERACIÓN 5

Anulación de órdenes de Pago Masiva	Anulación de Órdenes de Pago Puntual	Historia de Usuario	
	Codificación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
		Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
		Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
Programación en pareja y interacción con el cliente e integración continua		**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
		Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
	Refactorización	**Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN**	7
		Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
	Ejecución de simulación de pase	**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	-7
No obtiene correctamente el proceso, la cuenta		**Cada error cometido**	-5
Se corrigió la obtención del código de proceso y ejecución		**Cada error corregido**	+10
		Cada error no corregido	-5
		Cada error corregido pero que se deba mejorar	+7

Anulación de Consignaciones	Revertir saldo y cuota de devengado	Historia de Usuario	
Codificación	Codificación	**Los aspectos que se han realizado correctamente en cada ITERACIÓN**	+5
		Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10
		Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7
Programación en pareja y interacción con el cliente e integración continua	Programación en pareja y interacción con el cliente e integración continua	**Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente**	+3
		Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0
Generación de queries de reversión		**Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN**	7
		Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5
Generación de queries de reversión		**Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente**	-7
	No muestra el mensaje tanto en el archivo log como en la consola	**Cada error cometido**	-5
	Se corrigió el mensaje tanto en la consola como el archivo log	**Cada error corregido**	+10
		Cada error no corregido	-5
Registro de los reintegros cuando se anula la consignación		**Cada error corregido pero que se deba mejorar**	+7

Historia de Usuario	Puntaje	Puntaje TOTAL
Los aspectos que se han realizado correctamente en cada ITERACIÓN	+5	15	38
Un aspecto realizado correctamente en una ITERACIÓN que se realiza incorrectamente en una ITERACIÓN posterior	-10	0
Un aspecto realizado correctamente en una ITERACIÓN anterior y que en la ITERACIÓN actual se deba mejorar	-7	0
Los aspectos positivos que se mantienen de una ITERACIÓN al siguiente	+3	6
Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan	0	0
Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en la siguiente ITERACIÓN	7	14
Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en la ITERACIÓN siguiente.	-5	0
Un aspecto que se pueda mejorar y se convierta en un error en la ITERACIÓN siguiente	-7	-14
Cada error cometido	-5	-10
Cada error corregido	+10	20
Cada error no corregido	-5	0
Cada error corregido pero que se deba mejorar	+7	7

ANEXO 4

TÉRMINOS Y DEFINICIONES

Requerimientos	Parte de Requerimiento
SAR	Sistema de Administración de Requerimientos
PAS	Equipo de proyecto de la fábrica de software encargado de la administración de los sistemas para el cliente.
OTI-RT	Oficina de Tecnología de Información - Recursos Tecnológicos
CS	Coordinador de Sistemas de la División de Desarrollos Tecnológicos del cliente
GRS	Gestor de Requerimientos de Sistemas del cliente
UA	Usuario autorizado del cliente. Usuario facultado para solicitar y/o registrar requerimientos, por disposición del área dueña del sistema o por la División de Proyectos.

ANEXO 5

TAREAS DE RUP, SCRUM Y XP DE LA METODOLOGÍA PROPUESTA

Id	Nombre	FASE	ITERACIONES	METODOLOGÍA
1	SERVICIO DE MANTENIMIENTO Y DESARROLLO DE SISTEMAS DYM2011			
2	Línea L14_PAGO_03			
3	**Pagos**			
4	**Procesos de Gestión**			
5	Inicio del mantenimiento ciclo actual (11)			
6	Recepción de lista de requerimientos priorizados	GESTION	ITERACIÓN 1	SCRUM
7	Preparación de la Pila del Producto (Requerimientos priorizados)	GESTION	ITERACIÓN 1	SCRUM
8	Dar conformidad a la lista de requerimientos priorizados	GESTION	ITERACIÓN 1	SCRUM
9	**Elaboración del cronograma del nuevo ciclo de mantenimiento**	GESTION		
10	Realizar la reunión de planificación de las Iteraciones	GESTION	ITERACIÓN 1	SCRUM
11	Definir qué historias incluir en cada Iteración (Elaborar lista maestra de requerimiento)	GESTION	ITERACIÓN 2	SCRUM
12	Realizar la lista de historias en cada iteración (Pila de Iteraciones)	GESTION	ITERACIÓN 2	SCRUM
13	Estimar requerimientos	GESTION	ITERACIÓN 2	SCRUM
14	Elaboración y etiquetado del cronograma	GESTION	ITERACIÓN 2	RUP
15	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTION	ITERACIÓN 2	RUP
16	Aseguramiento de calidad y etiquetado del cronograma	GESTION	ITERACIÓN 2	RUP
17	Revisión y Envío de Cronograma	GESTION	ITERACIÓN 2	RUP
18	Revisa y Aprueba Cronograma	GESTION	ITERACIÓN 2	RUP
19	Actualiza matriz de control de versiones de cronogramas aprobados	GESTION	ITERACIÓN 2	SCRUM
20	Cierre de mantenimiento anterior (10)	GESTION		
21	Elaborar informe de cierre de ciclo de producción	GESTION	ITERACIÓN 3	SCRUM
22	Elaborar y revisar el relatorio del Ciclo de Producción	GESTION	ITERACIÓN 3	RUP
23	Mantener parámetros de estimación	GESTION	ITERACIÓN 3	SCRUM
24	Revisión de QA	GESTION	ITERACIÓN 3	RUP
25	Actividades de gestión de la configuración al cierre de ciclo	GESTION		
26	Auditar gestión de configuración de MA (Ciclo anterior)	GESTION	ITERACIÓN 3	RUP
27	Seguimiento del mantenimiento (11)	GESTION		
28	**Semana 1**	GESTION		

29	Asignación de Trabajo	GESTIÓN	ITERACIÓN 4	SCRUM
30	Reuniones de Comité Interno	GESTIÓN		
31	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN		
32	Recolección de medidas y Generación del Tablero de Control (Línea)	GESTIÓN	ITERACIÓN 4	RUP
33	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
34	Realizar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
35	Elaborar Acta de Reunión Interna	GESTIÓN	ITERACIÓN 4	SCRUM
36	Reuniones de Comité Interno de Sistemas	GESTIÓN		
37	Realizar Comité de Analistas de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
38	Reuniones de Comité de Línea (AS+CLM+ONP)	GESTIÓN		
39	Preparar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
40	Realizar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
41	Reprogramación de Cronograma	GESTIÓN		
42	Reunión de Pre-Incepción	GESTIÓN	ITERACIÓN 4	RUP
43	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 4	RUP
44	Estimar requerimientos	GESTIÓN	ITERACIÓN 4	SCRUM
45	Actualización y etiquetado de cronograma	GESTIÓN	ITERACIÓN 4	RUP
46	Realizar las Demostraciones o Revisiones de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
47	Realizar las Retrospectivas de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
48	Informar Cuadro de Reprogramaciones	GESTIÓN	ITERACIÓN 4	SCRUM
49	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 4	RUP
50	Aseguramiento de Calidad del Cronograma	GESTIÓN	ITERACIÓN 4	RUP
51	Envío del cronograma	GESTIÓN	ITERACIÓN 4	RUP
52	Aprobación del cronograma	GESTIÓN	ITERACIÓN 4	RUP
53	**Semana 2**	GESTIÓN		
54	Asignación de Trabajo	GESTIÓN	ITERACIÓN 4	SCRUM
55	Reuniones de Comité Interno	GESTIÓN		
56	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN		
57	Recolección de medidas y Generación del Tablero de Control (Línea)	GESTIÓN	ITERACIÓN 4	RUP
58	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
59	Realizar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
60	Elaborar Acta de Reunión Interna	GESTIÓN	ITERACIÓN 4	SCRUM
61	Reuniones de Comité Interno de Sistemas	GESTIÓN		
62	Realizar Comité de Analistas de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM

63	Reuniones de Comité de Línea (AS+CLM+ONP)	GESTIÓN		
64	Preparar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
65	Realizar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
66	Reprogramación de Cronograma	GESTIÓN		
67	Reunión de Pre-Incepción	GESTIÓN	ITERACIÓN 4	RUP
68	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 4	RUP
69	Estimar requerimientos	GESTIÓN	ITERACIÓN 4	SCRUM
70	Actualización y etiquetado de cronograma	GESTIÓN	ITERACIÓN 4	RUP
71	Realizar las Demostraciones o Revisiones de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
72	Realizar las Retrospectivas de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
73	Informar Cuadro de Reprogramaciones	GESTIÓN	ITERACIÓN 4	SCRUM
74	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 4	RUP
75	Aseguramiento de Calidad del Cronograma	GESTIÓN	ITERACIÓN 4	RUP
76	Envío del cronograma	GESTIÓN	ITERACIÓN 4	RUP
77	Aprobación del cronograma	GESTIÓN	ITERACIÓN 4	RUP
78	**Semana 3**	GESTIÓN		
79	Asignación de Trabajo	GESTIÓN	ITERACIÓN 4	SCRUM
80	Reuniones de Comité Interno	GESTIÓN		
81	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN		
82	Recolección de medidas y Generación del Tablero de Control (Línea)	GESTIÓN	ITERACIÓN 4	RUP
83	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
84	Realizar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
85	Elaborar Acta de Reunión Interna	GESTIÓN	ITERACIÓN 4	SCRUM
86	Reuniones de Comité Interno de Sistemas	GESTIÓN		
87	Realizar Comité de Analistas de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
88	Reuniones de Comité de Línea (AS+CLM+ONP)	GESTIÓN		
89	Preparar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
90	Realizar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
91	Reprogramación de Cronograma	GESTIÓN		
92	Reunión de Pre-Incepción	GESTIÓN	ITERACIÓN 4	RUP
93	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 4	RUP
94	Estimar requerimientos	GESTIÓN	ITERACIÓN 4	SCRUM
95	Actualización y etiquetado de cronograma	GESTIÓN	ITERACIÓN 4	RUP
96	Realizar las Demostraciones o Revisiones de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM

97	Realizar las Retrospectivas de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
98	Informar Cuadro de Reprogramaciones	GESTIÓN	ITERACIÓN 4	SCRUM
99	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 4	RUP
100	Aseguramiento de Calidad del Cronograma	GESTIÓN	ITERACIÓN 4	RUP
101	Envío del cronograma	GESTIÓN	ITERACIÓN 4	RUP
102	Aprobación del cronograma	GESTIÓN	ITERACIÓN 4	RUP
103	**Semana 4**	GESTIÓN		
104	Asignación de Trabajo	GESTIÓN	ITERACIÓN 4	SCRUM
105	Reuniones de Comité Interno	GESTIÓN		
106	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN		
107	Recolección de medidas y Generación del Tablero de Control (Línea)	GESTIÓN	ITERACIÓN 4	RUP
108	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
109	Realizar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
110	Elaborar Acta de Reunión Interna	GESTIÓN	ITERACIÓN 4	SCRUM
111	Reuniones de Comité Interno de Sistemas	GESTIÓN		
112	Realizar Comité de Analistas de Mantenimiento	GESTIÓN	ITERACIÓN 4	SCRUM
113	Reuniones de Comité de Línea (AS+CLM+ONP)	GESTIÓN		
114	Preparar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
115	Realizar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 4	RUP
116	Reprogramación de Cronograma	GESTIÓN		
117	Reunión de Pre-Incepción	GESTIÓN	ITERACIÓN 4	RUP
118	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 4	RUP
119	Estimar requerimientos	GESTIÓN	ITERACIÓN 4	SCRUM
120	Actualización y etiquetado de cronograma	GESTIÓN	ITERACIÓN 4	RUP
121	Realizar las Demostraciones o Revisiones de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
122	Realizar las Retrospectivas de la Iteración	GESTIÓN	ITERACIÓN 4	SCRUM
123	Informar Cuadro de Reprogramaciones	GESTIÓN	ITERACIÓN 4	SCRUM
124	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 4	RUP
125	Aseguramiento de Calidad del Cronograma	GESTIÓN	ITERACIÓN 4	RUP
126	Envío del cronograma	GESTIÓN	ITERACIÓN 4	RUP
127	Aprobación del cronograma	GESTIÓN	ITERACIÓN 4	RUP
128	**Semana 5**	GESTIÓN		
129	Asignación de Trabajo	GESTIÓN	ITERACIÓN 5	SCRUM
130	Reuniones de Comité Interno	GESTIÓN		
131	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN		

132	Recolección de medidas y Generación del Tablero de Control (Línea)	GESTIÓN	ITERACIÓN 5	RUP
133	Preparar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 5	RUP
134	Realizar Comité Interno de Línea de Mantenimiento	GESTIÓN	ITERACIÓN 5	SCRUM
135	Elaborar Acta de Reunión Interna	GESTIÓN	ITERACIÓN 5	SCRUM
136	Reuniones de Comité Interno de Sistemas	GESTIÓN		
137	Realizar Comité de Analistas de Mantenimiento	GESTIÓN	ITERACIÓN 5	RUP
138	Reuniones de Comité de Línea (AS+CLM+ONP)	GESTIÓN		
139	Preparar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 5	RUP
140	Realizar Comité de Sistemas por Línea de Mantenimiento	GESTIÓN	ITERACIÓN 5	SCRUM
141	Reprogramación de Cronograma	GESTIÓN		
142	Reunión de Pre-Incepción	GESTIÓN	ITERACIÓN 5	SCRUM
143	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 5	RUP
144	Estimar requerimientos	GESTIÓN	ITERACIÓN 5	SCRUM
145	Actualización y etiquetado de cronograma	GESTIÓN	ITERACIÓN 5	RUP
146	Realizar las Demostraciones o Revisiones de la Iteración	GESTIÓN	ITERACIÓN 5	SCRUM
147	Realizar las Retrospectivas de la Iteración	GESTIÓN	ITERACIÓN 5	SCRUM
148	Informar Cuadro de Reprogramaciones	GESTIÓN	ITERACIÓN 5	SCRUM
149	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 5	RUP
150	Aseguramiento de Calidad del Cronograma	GESTIÓN	ITERACIÓN 5	RUP
151	Envío del cronograma	GESTIÓN	ITERACIÓN 5	RUP
152	Aprobación del cronograma	GESTIÓN	ITERACIÓN 5	RUP
153	**Inicio del mantenimiento ciclo siguiente (12)**	GESTIÓN		
154	Recepción de lista de requerimientos priorizados	GESTIÓN	ITERACIÓN 5	SCRUM
155	Preparación de la Pila del Producto (Requerimientos priorizados)	GESTIÓN	ITERACIÓN 5	SCRUM
156	Dar conformidad a la lista de requerimientos priorizados	GESTIÓN	ITERACIÓN 5	SCRUM
157	Elaboración del cronograma del nuevo ciclo de mantenimiento	GESTIÓN		
158	Realizar la reunión de planificación de las Iteraciones	GESTIÓN	ITERACIÓN 5	SCRUM
159	Definir qué historias incluir en cada Iteración (Elaborar lista maestra de requerimientos)	GESTIÓN	ITERACIÓN 5	SCRUM
160	Realizar la lista de historias en cada iteración (Pila de Iteraciones)	GESTIÓN	ITERACIÓN 5	SCRUM
161	Reunión de Pre-análisis	GESTIÓN	ITERACIÓN 5	SCRUM
162	Elaborar lista maestra de requerimiento	GESTIÓN	ITERACIÓN 5	RUP
163	Estimar requerimientos	GESTIÓN	ITERACIÓN 5	SCRUM

164	Elaboración y etiquetado del cronograma	GESTIÓN	ITERACIÓN 5	RUP
165	Auditoria de Gestión de Configuración de cronograma (auditoria semanal)	GESTIÓN	ITERACIÓN 5	RUP
166	Aseguramiento de calidad y etiquetado del cronograma	GESTIÓN	ITERACIÓN 5	RUP
167	Revisión y Envío de Cronograma	GESTIÓN	ITERACIÓN 5	RUP
168	Revisa y Aprueba Cronograma	GESTIÓN	ITERACIÓN 5	RUP
169	Actualiza matriz de control de versiones de cronogramas aprobados	GESTIÓN	ITERACIÓN 5	RUP
170	**Procesos de Ingeniería**			
171	Actividades del Ciclo Actual CMMI			
172	Elaboración de requerimientos			
173	requerimientos # 623 (NSP) - MODIFICAR DEPURACION DE FALLECIDOS Y ANULACION DE PAGOS PARA QUE CONSIDERE PLANILLAS ADICIONALES			
174	**Incepción – Pre-análisis**			
175	Reunión de Pre-análisis	INCEPCIÓN		RUP
176	Elaboración del Pre-análisis del Requerimiento	INCEPCIÓN		RUP
177	Verificación del Pre-análisis del Requerimiento	INCEPCIÓN		RUP
178	**Revisión de Pares**	INCEPCIÓN		
179	Ejecución de la Revisión de Pares	INCEPCIÓN		RUP
180	**Gestión de Configuración**	INCEPCIÓN		
181	Solicitar auditoria de Gestión de Configuración	INCEPCIÓN		RUP
182	Auditoria de Gestión de Configuración MA	INCEPCIÓN		RUP
183	Envío de Documento de Pre-análisis	INCEPCIÓN		RUP
184	Aprobación del Pre-análisis por parte del UA	INCEPCIÓN		RUP
185	Aprobación del Pre-análisis por parte del GRS	INCEPCIÓN		RUP
186	Ajuste del documento de Pre-análisis	INCEPCIÓN		RUP
187	**Elaboración - Análisis**	ELABORACIÓN		
188	Escribir historias de usuario	ELABORACIÓN		XP
189	Probar tecnologías a utilizar	ELABORACIÓN		XP
190	Estimar esfuerzo para historias de usuario	ELABORACIÓN		XP
191	Análisis y Elaboración de Documento de Análisis - Sección Funcional	ELABORACIÓN		RUP
192	Verificación del Documento de Análisis	ELABORACIÓN		RUP
193	Reescribir las historias de usuario	ELABORACIÓN		XP
194	Formular el plan de versiones	ELABORACIÓN		XP
195	Revisión de Pares	ELABORACIÓN		
196	Ejecución de la Revisión de Pares	ELABORACIÓN		RUP
197	Gestión de Configuración	ELABORACIÓN		
198	Solicitar auditoria de Gestión de Configuración	ELABORACIÓN		RUP
199	Auditoria de Gestión de Configuración	ELABORACIÓN		RUP

	MA			
200	Envío de Documento de Análisis	ELABORACIÓN		RUP
201	Solicitud de Ajuste de Análisis	ELABORACIÓN		RUP
202	Envío de Documento de Análisis	ELABORACIÓN		RUP
203	Revisión y Aprobación de Documento de Análisis - CS	ELABORACIÓN		RUP
204	Revisión y Aprobación de Documento de Análisis - UA	ELABORACIÓN		RUP
205	Revisión y Aprobación de Documento de Análisis - GRS	ELABORACIÓN		RUP
206	**Construcción Iteración 1**	CONSTRUCCIÓN	ITERACIÓN 1	
207	Desarrollo de la Solución	CONSTRUCCIÓN	ITERACIÓN 1	
208	Definir la arquitectura técnica	CONSTRUCCIÓN	ITERACIÓN 1	XP
209	Escribir tareas de ingeniería	CONSTRUCCIÓN	ITERACIÓN 1	XP
210	Formular el plan de iteraciones	CONSTRUCCIÓN	ITERACIÓN 1	XP
211	Crear pruebas de aceptación	CONSTRUCCIÓN	ITERACIÓN 1	XP
212	Implementar las interfases	CONSTRUCCIÓN	ITERACIÓN 1	XP
213	Escribir tarjetas CRC para cada tarea de ingeniería	CONSTRUCCIÓN	ITERACIÓN 1	XP
214	Implementar la base de datos física	CONSTRUCCIÓN	ITERACIÓN 1	XP
215	Crear pruebas unitarias para las clases control	CONSTRUCCIÓN	ITERACIÓN 1	XP
216	Implementar Modificaciones	CONSTRUCCIÓN	ITERACIÓN 1	XP
217	Actualización de Harvest	CONSTRUCCIÓN	ITERACIÓN 1	RUP
218	Elaboración de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 1	RUP
219	Pruebas Unitarias	CONSTRUCCIÓN	ITERACIÓN 1	RUP
220	Realizar integración continua	CONSTRUCCIÓN	ITERACIÓN 1	RUP
221	Ejecutar pruebas de integración para una historia de usuario	CONSTRUCCIÓN	ITERACIÓN 1	RUP
222	Verificación de Documento de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 1	RUP
223	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 1	RUP
224	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 1	RUP
225	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
226	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 1	RUP
227	Ejecutar la Revisión de QA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
228	Envío de los casos de Pruebas	CONSTRUCCIÓN	ITERACIÓN 1	RUP
229	Aprobación de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 1	RUP
230	Pruebas Internas	CONSTRUCCIÓN	ITERACIÓN 1	
231	Elaboración de data de prueba	CONSTRUCCIÓN	ITERACIÓN 1	
232	Pruebas funcionales	CONSTRUCCIÓN	ITERACIÓN 1	RUP
233	Pruebas técnicas	CONSTRUCCIÓN	ITERACIÓN 1	RUP
234	Llenar Checklist (Analista y Programador)	CONSTRUCCIÓN	ITERACIÓN 1	RUP
235	Elaborar documento de Pase a QA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
236	Actualizar matriz de trazabilidad	CONSTRUCCIÓN	ITERACIÓN 1	RUP
237	Solicitar QC	CONSTRUCCIÓN	ITERACIÓN 1	RUP
238	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 1	
239	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 1	RUP

240	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
241	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 1	
242	Revisar Checklist de Ingeniería Mantenimiento	CONSTRUCCIÓN	ITERACIÓN 1	RUP
243	Revisión Técnica	CONSTRUCCIÓN	ITERACIÓN 1	RUP
244	Preparar Ambiente de Pruebas	CONSTRUCCIÓN	ITERACIÓN 1	RUP
245	Revisión Funcional	CONSTRUCCIÓN	ITERACIÓN 1	RUP
246	Solicitar Pase a QA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
247	Autoriza Pase a QA	CONSTRUCCIÓN	ITERACIÓN 1	RUP
248	Pruebas Funcionales y Sistemas	CONSTRUCCIÓN	ITERACIÓN 1	
249	Verificar ejecución de pase a QA/Producción	CONSTRUCCIÓN	ITERACIÓN 1	RUP
250	Pruebas funcionales y aceptación	CONSTRUCCIÓN	ITERACIÓN 1	RUP
251	Actualización de Documento de PPFF	CONSTRUCCIÓN	ITERACIÓN 1	RUP
252	Aprobación Pruebas Funcionales	CONSTRUCCIÓN	ITERACIÓN 1	RUP
253	Pruebas de sistemas y aceptación	CONSTRUCCIÓN	ITERACIÓN 1	RUP
254	**Construcción Iteración 2**	CONSTRUCCIÓN	ITERACIÓN 2	
255	Desarrollo de la Solución	CONSTRUCCIÓN	ITERACIÓN 2	
256	Definir la arquitectura técnica	CONSTRUCCIÓN	ITERACIÓN 2	XP
257	Escribir tareas de ingeniería	CONSTRUCCIÓN	ITERACIÓN 2	XP
258	Formular el plan de iteraciones	CONSTRUCCIÓN	ITERACIÓN 2	XP
259	Crear pruebas de aceptación	CONSTRUCCIÓN	ITERACIÓN 2	XP
260	Implementar las interfases	CONSTRUCCIÓN	ITERACIÓN 2	XP
261	Escribir tarjetas CRC para cada tarea de ingeniería	CONSTRUCCIÓN	ITERACIÓN 2	XP
262	Implementar la base de datos física	CONSTRUCCIÓN	ITERACIÓN 2	XP
263	Crear pruebas unitarias para las clases control	CONSTRUCCIÓN	ITERACIÓN 2	XP
264	Implementar Modificaciones	CONSTRUCCIÓN	ITERACIÓN 2	XP
265	Actualización de Harvest	CONSTRUCCIÓN	ITERACIÓN 2	RUP
266	Elaboración de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 2	RUP
267	Pruebas Unitarias	CONSTRUCCIÓN	ITERACIÓN 2	RUP
268	Realizar integración continua	CONSTRUCCIÓN	ITERACIÓN 2	RUP
269	Ejecutar pruebas de integración para una historia de usuario	CONSTRUCCIÓN	ITERACIÓN 2	RUP
270	Verificación de Documento de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 2	RUP
271	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 2	RUP
272	Solicitar auditoría de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 2	RUP
273	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
274	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 2	RUP
275	Ejecutar la Revisión de QA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
276	Envío de los casos de Pruebas	CONSTRUCCIÓN	ITERACIÓN 2	RUP
277	Aprobación de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 2	RUP
278	Pruebas Internas	CONSTRUCCIÓN	ITERACIÓN 2	
279	Elaboración de data de prueba	CONSTRUCCIÓN	ITERACIÓN 2	
280	Pruebas funcionales	CONSTRUCCIÓN	ITERACIÓN 2	RUP
281	Pruebas técnicas	CONSTRUCCIÓN	ITERACIÓN 2	RUP

282	Llenar Checklist (Analista y Programador)	CONSTRUCCIÓN	ITERACIÓN 2	RUP
283	Elaborar documento de Pase a QA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
284	Actualizar matriz de trazabilidad	CONSTRUCCIÓN	ITERACIÓN 2	RUP
285	Solicitar QC	CONSTRUCCIÓN	ITERACIÓN 2	RUP
286	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 2	
287	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 2	RUP
288	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
289	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 2	
290	Revisar Checklist de Ingeniería Mantenimiento	CONSTRUCCIÓN	ITERACIÓN 2	RUP
291	Revisión Técnica	CONSTRUCCIÓN	ITERACIÓN 2	RUP
292	Preparar Ambiente de Pruebas	CONSTRUCCIÓN	ITERACIÓN 2	RUP
293	Revisión Funcional	CONSTRUCCIÓN	ITERACIÓN 2	RUP
294	Solicitar Pase a QA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
295	Autoriza Pase a QA	CONSTRUCCIÓN	ITERACIÓN 2	RUP
296	Pruebas Funcionales y Sistemas	CONSTRUCCIÓN	ITERACIÓN 2	
297	Verificar ejecución de pase a QA/Producción	CONSTRUCCIÓN	ITERACIÓN 2	RUP
298	Pruebas funcionales y aceptación	CONSTRUCCIÓN	ITERACIÓN 2	RUP
299	Actualización de Documento de PPFF	CONSTRUCCIÓN	ITERACIÓN 2	RUP
300	Aprobación Pruebas Funcionales	CONSTRUCCIÓN	ITERACIÓN 2	RUP
301	Pruebas de sistemas y aceptación	CONSTRUCCIÓN	ITERACIÓN 2	RUP
302	**Construcción Iteración 3**	CONSTRUCCIÓN	ITERACIÓN 3	
303	Desarrollo de la Solución	CONSTRUCCIÓN	ITERACIÓN 3	
304	Definir la arquitectura técnica	CONSTRUCCIÓN	ITERACIÓN 3	XP
305	Escribir tareas de ingeniería	CONSTRUCCIÓN	ITERACIÓN 3	XP
306	Formular el plan de iteraciones	CONSTRUCCIÓN	ITERACIÓN 3	XP
307	Crear pruebas de aceptación	CONSTRUCCIÓN	ITERACIÓN 3	XP
308	Implementar las interfases	CONSTRUCCIÓN	ITERACIÓN 3	XP
309	Escribir tarjetas CRC para cada tarea de ingeniería	CONSTRUCCIÓN	ITERACIÓN 3	XP
310	Implementar la base de datos física	CONSTRUCCIÓN	ITERACIÓN 3	XP
311	Crear pruebas unitarias para las clases control	CONSTRUCCIÓN	ITERACIÓN 3	XP
312	Implementar Modificaciones	CONSTRUCCIÓN	ITERACIÓN 3	XP
313	Actualización de Harvest	CONSTRUCCIÓN	ITERACIÓN 3	RUP
314	Elaboración de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 3	RUP
315	Pruebas Unitarias	CONSTRUCCIÓN	ITERACIÓN 3	RUP
316	Realizar integración continua	CONSTRUCCIÓN	ITERACIÓN 3	RUP
317	Ejecutar pruebas de integración para una historia de usuario	CONSTRUCCIÓN	ITERACIÓN 3	RUP
318	Verificación de Documento de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 3	RUP
319	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 3	RUP
320	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 3	RUP
321	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
322	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 3	RUP

323	Ejecutar la Revisión de QA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
324	Envío de los casos de Pruebas	CONSTRUCCIÓN	ITERACIÓN 3	RUP
325	Aprobación de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 3	RUP
326	Pruebas Internas	CONSTRUCCIÓN	ITERACIÓN 3	
327	Elaboración de data de prueba	CONSTRUCCIÓN	ITERACIÓN 3	
328	Pruebas funcionales	CONSTRUCCIÓN	ITERACIÓN 3	RUP
329	Pruebas técnicas	CONSTRUCCIÓN	ITERACIÓN 3	RUP
330	Llenar Checklist (Analista y Programador)	CONSTRUCCIÓN	ITERACIÓN 3	RUP
331	Elaborar documento de Pase a QA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
332	Actualizar matriz de trazabilidad	CONSTRUCCIÓN	ITERACIÓN 3	RUP
333	Solicitar QC	CONSTRUCCIÓN	ITERACIÓN 3	RUP
334	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 3	
335	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 3	RUP
336	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
337	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 3	
338	Revisar Checklist de Ingeniería Mantenimiento	CONSTRUCCIÓN	ITERACIÓN 3	RUP
339	Revisión Técnica	CONSTRUCCIÓN	ITERACIÓN 3	RUP
340	Preparar Ambiente de Pruebas	CONSTRUCCIÓN	ITERACIÓN 3	RUP
341	Revisión Funcional	CONSTRUCCIÓN	ITERACIÓN 3	RUP
342	Solicitar Pase a QA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
343	Autoriza Pase a QA	CONSTRUCCIÓN	ITERACIÓN 3	RUP
344	Pruebas Funcionales y Sistemas	CONSTRUCCIÓN	ITERACIÓN 3	
345	Verificar ejecución de pase a QA/Producción	CONSTRUCCIÓN	ITERACIÓN 3	RUP
346	Pruebas funcionales y aceptación	CONSTRUCCIÓN	ITERACIÓN 3	RUP
347	Actualización de Documento de PPFF	CONSTRUCCIÓN	ITERACIÓN 3	RUP
348	Aprobación Pruebas Funcionales	CONSTRUCCIÓN	ITERACIÓN 3	RUP
349	Pruebas de sistemas y aceptación	CONSTRUCCIÓN	ITERACIÓN 3	RUP
350	**Construcción Iteración 4**	CONSTRUCCIÓN	ITERACIÓN 4	
351	Desarrollo de la Solución	CONSTRUCCIÓN	ITERACIÓN 4	
352	Definir la arquitectura técnica	CONSTRUCCIÓN	ITERACIÓN 4	XP
353	Escribir tareas de ingeniería	CONSTRUCCIÓN	ITERACIÓN 4	XP
354	Formular el plan de iteraciones	CONSTRUCCIÓN	ITERACIÓN 4	XP
355	Crear pruebas de aceptación	CONSTRUCCIÓN	ITERACIÓN 4	XP
356	Implementar las interfases	CONSTRUCCIÓN	ITERACIÓN 4	XP
357	Escribir tarjetas CRC para cada tarea de ingeniería	CONSTRUCCIÓN	ITERACIÓN 4	XP
358	Implementar la base de datos física	CONSTRUCCIÓN	ITERACIÓN 4	XP
359	Crear pruebas unitarias para las clases control	CONSTRUCCIÓN	ITERACIÓN 4	XP
360	Implementar Modificaciones	CONSTRUCCIÓN	ITERACIÓN 4	XP
361	Actualización de Sistema de Configuración de Versiones	CONSTRUCCIÓN	ITERACIÓN 4	RUP
362	Elaboración de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 4	RUP
363	Pruebas Unitarias	CONSTRUCCIÓN	ITERACIÓN 4	RUP
364	Realizar integración continua	CONSTRUCCIÓN	ITERACIÓN 4	RUP
365	Ejecutar pruebas de integración para	CONSTRUCCIÓN	ITERACIÓN 4	RUP

	una historia de usuario			
366	Verificación de Documento de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 4	RUP
367	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 4	RUP
368	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 4	RUP
369	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
370	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 4	RUP
371	Ejecutar la Revisión de QA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
372	Envío de los casos de Pruebas	CONSTRUCCIÓN	ITERACIÓN 4	RUP
373	Aprobación de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 4	RUP
374	Pruebas Internas	CONSTRUCCIÓN	ITERACIÓN 4	
375	Elaboración de data de prueba	CONSTRUCCIÓN	ITERACIÓN 4	
376	Pruebas funcionales	CONSTRUCCIÓN	ITERACIÓN 4	RUP
377	Pruebas técnicas	CONSTRUCCIÓN	ITERACIÓN 4	RUP
378	Llenar Checklist (Analista y Programador)	CONSTRUCCIÓN	ITERACIÓN 4	RUP
379	Elaborar documento de Pase a QA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
380	Actualizar matriz de trazabilidad	CONSTRUCCIÓN	ITERACIÓN 4	RUP
381	Solicitar QC	CONSTRUCCIÓN	ITERACIÓN 4	RUP
382	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 4	
383	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 4	RUP
384	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
385	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 4	
386	Revisar Checklist de Ingeniería Mantenimiento	CONSTRUCCIÓN	ITERACIÓN 4	RUP
387	Revisión Técnica	CONSTRUCCIÓN	ITERACIÓN 4	RUP
388	Preparar Ambiente de Pruebas	CONSTRUCCIÓN	ITERACIÓN 4	RUP
389	Revisión Funcional	CONSTRUCCIÓN	ITERACIÓN 4	RUP
390	Solicitar Pase a QA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
391	Autoriza Pase a QA	CONSTRUCCIÓN	ITERACIÓN 4	RUP
392	Pruebas Funcionales y Sistemas	CONSTRUCCIÓN	ITERACIÓN 4	
393	Verificar ejecución de pase a QA/Producción	CONSTRUCCIÓN	ITERACIÓN 4	RUP
394	Pruebas funcionales y aceptación	CONSTRUCCIÓN	ITERACIÓN 4	RUP
395	Actualización de Documento de PPFF	CONSTRUCCIÓN	ITERACIÓN 4	RUP
396	Aprobación Pruebas Funcionales	CONSTRUCCIÓN	ITERACIÓN 4	RUP
397	Pruebas de sistemas y aceptación	CONSTRUCCIÓN	ITERACIÓN 4	RUP
398	**Construcción Iteración 5**	CONSTRUCCIÓN	ITERACIÓN 5	
399	Desarrollo de la Solución	CONSTRUCCIÓN	ITERACIÓN 5	
400	Definir la arquitectura técnica	CONSTRUCCIÓN	ITERACIÓN 5	XP
401	Escribir tareas de ingeniería	CONSTRUCCIÓN	ITERACIÓN 5	XP
402	Formular el plan de iteraciones	CONSTRUCCIÓN	ITERACIÓN 5	XP
403	Crear pruebas de aceptación	CONSTRUCCIÓN	ITERACIÓN 5	XP
404	Implementar las interfases	CONSTRUCCIÓN	ITERACIÓN 5	XP
405	Escribir tarjetas CRC para cada tarea de ingeniería	CONSTRUCCIÓN	ITERACIÓN 5	XP
406	Implementar la base de datos física	CONSTRUCCIÓN	ITERACIÓN 5	XP

407	Crear pruebas unitarias para las clases control	CONSTRUCCIÓN	ITERACIÓN 5	XP
408	Implementar Modificaciones	CONSTRUCCIÓN	ITERACIÓN 5	XP
409	Actualización de Harvest	CONSTRUCCIÓN	ITERACIÓN 5	RUP
410	Elaboración de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 5	RUP
411	Pruebas Unitarias	CONSTRUCCIÓN	ITERACIÓN 5	RUP
412	Realizar integración continua	CONSTRUCCIÓN	ITERACIÓN 5	RUP
413	Ejecutar pruebas de integración para una historia de usuario	CONSTRUCCIÓN	ITERACIÓN 5	RUP
414	Verificación de Documento de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 5	RUP
415	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 5	RUP
416	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 5	RUP
417	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
418	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 5	RUP
419	Ejecutar la Revisión de QA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
420	Envío de los casos de Pruebas	CONSTRUCCIÓN	ITERACIÓN 5	RUP
421	Aprobación de Casos de Prueba	CONSTRUCCIÓN	ITERACIÓN 5	RUP
422	Pruebas Internas	CONSTRUCCIÓN	ITERACIÓN 5	
423	Elaboración de data de prueba	CONSTRUCCIÓN	ITERACIÓN 5	
424	Pruebas funcionales	CONSTRUCCIÓN	ITERACIÓN 5	RUP
425	Pruebas técnicas	CONSTRUCCIÓN	ITERACIÓN 5	RUP
426	Llenar Checklist (Analista y Programador)	CONSTRUCCIÓN	ITERACIÓN 5	RUP
427	Elaborar documento de Pase a QA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
428	Actualizar matriz de trazabilidad	CONSTRUCCIÓN	ITERACIÓN 5	RUP
429	Solicitar QC	CONSTRUCCIÓN	ITERACIÓN 5	RUP
430	Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 5	
431	Solicitar auditoria de Gestión de Configuración	CONSTRUCCIÓN	ITERACIÓN 5	RUP
432	Auditoria de Gestión de Configuración MA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
433	Aseguramiento de calidad	CONSTRUCCIÓN	ITERACIÓN 5	
434	Revisar Checklist de Ingeniería Mantenimiento	CONSTRUCCIÓN	ITERACIÓN 5	RUP
435	Revisión Técnica	CONSTRUCCIÓN	ITERACIÓN 5	RUP
436	Preparar Ambiente de Pruebas	CONSTRUCCIÓN	ITERACIÓN 5	RUP
437	Revisión Funcional	CONSTRUCCIÓN	ITERACIÓN 5	RUP
438	Solicitar Pase a QA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
439	Autoriza Pase a QA	CONSTRUCCIÓN	ITERACIÓN 5	RUP
440	Pruebas Funcionales y Sistemas	CONSTRUCCIÓN	ITERACIÓN 5	
441	Verificar ejecución de pase a QA/Producción	CONSTRUCCIÓN	ITERACIÓN 5	RUP
442	Pruebas funcionales y aceptación	CONSTRUCCIÓN	ITERACIÓN 5	RUP
443	Actualización de Documento de PPFF	CONSTRUCCIÓN	ITERACIÓN 5	RUP
444	Aprobación Pruebas Funcionales	CONSTRUCCIÓN	ITERACIÓN 5	RUP
445	Pruebas de sistemas y aceptación	CONSTRUCCIÓN	ITERACIÓN 5	RUP
446	**Transición**	TRANSICION		
447	Pase a Producción	TRANSICION		

448	Solicitud de Pase a Producción - Sistema de Configuración de Versiones	**TRANSICION**		RUP
449	Ejecución de Pase a Producción	**TRANSICION**		RUP
450	Capacitación y Documentación de Requerimientos	**TRANSICION**		
451	PR # 623 (NSP) - MODIFICAR DEPURACION DE FALLECIDOS Y ANULACION DE PAGOS PARA QUE CONSIDERE PLANILLAS ADICIONALES	**TRANSICION**		
452	Actualizar Manual de Usuario	**TRANSICION**		RUP
453	Actualizar Manual de Sistemas	**TRANSICION**		RUP
454	Verificar actualización de Manuales	**TRANSICION**		RUP
455	Aprobación de Manuales	**TRANSICION**		RUP