UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MEJORAMIENTO DE LA GESTIÓN DE PROBLEMAS Y MANTENCIÓN DE SOFTWARE EN UNA EMPRESA DE SERVICIOS ELECTRÓNICOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN TECNOLOGÍAS DE INFORMACIÓN

ÁLVARO GONZALO MILLALÉN ÑANCULEO

PROFESOR GUÍA:
JOSÉ ALBERTO PINO URTUBIA

MIEMBROS DE LA COMISIÓN:
ALEXANDRE BERGEL
AGUSTÍN VILLENA MOYA
YADRAN ETEROVIC SOLANO

SANTIAGO DE CHILE
2014

RESUMEN

El desarrollo de esta tesis consistió en generar mejoras a los procesos de gestión de problemas y mantención de software de Acepta.com. El objetivo fue cumplir con la necesidad de corregir y evolucionar los productos de software en tiempos razonables, y de esta manera brindar una escalabilidad mayor a los servicios prestados.

El problema que se pretende resolver es la solución oportuna de defectos del software utilizado para bridar los servicios de la empresa.

Actualmente la empresa cuenta con más de 20 sistemas, que brindan servicios de facturación electrónica y certificación electrónica a más de 1500 empresas. Estos sistemas procesan más de 100 millones documentos al año, y de ellos depende en gran medida la situación tributaria de las empresas que utilizan el servicio. En este sentido, la atención oportuna de desperfectos en el software que brinda el servicio es crucial.

El propósito de esta tesis ha sido abordar este problema diseñando, e implementando el ciclo completo de solución de problemas del cliente. Esto supone la mejora y rediseño de los procesos actuales, integrando el proceso de gestión de problemas, y el de mantención de software.

El proceso resultante se propone apoyarlo con TI de modo de agilizar la coordinación de las tareas, la transición entre las diferentes fases, y la administración del personal a cargo de éstas.

El producto final de esta tesis es la identificación e implementación de nuevas prácticas, la formalización de procesos de transferencia y mantención del software, la definición de una organización de mantención, y la propuesta de un nuevo proceso de gestión de problemas con una mirada integradora. En su conjunto, este producto da una solución práctica al problema de evolución del software base de los servicios de esta empresa.

AGRADECIMIENTOS

Ha sido un período no excento de complicaciones. Donde la guía, el consejo, y el apoyo han sido fundamentales.

Quiero agradecer a quienes me brindaron su consejo o una palabra de aliento cuando lo necesité. Gracias por levantarme en esos momento de flaqueza.

Agradecer especialmente a mis padres. Su amor y ejemplo de superación me inspiran día a día en mi caminar por la vida.

Agradecer a aquellos profesores que me formaron en lo humano y profesional. Este logro también es de ellos.

Finalmente agradecer a mi profesor guía José Alberto Pino, y a los miembros de la comisión. Sin sus sabios consejos no hubiera sido posible llegar a buen termino este trabajo.

Álvaro Millalén Ñanculeo

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO 1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

Acepta es una empresa dedicada a brindar servicios de firma, generación y custodia de documentos electrónicos, y servicios de certificación electrónica de acuerdo a las normativas y leyes chilenas desde 1999 (ver anexo A).

Actualmente Acepta presta servicios de factura electrónica a más de 1500 empresas de todo el país, para lo cual se cuenta con un total de 22 sistemas, que en su totalidad suman 450 mil líneas de código.

El incremento en el volumen de clientes y carga de los sistemas, se ha abordado con un incremento en los recursos humanos involucrados en la prestación de los servicios y en help desk. Sin embargo, no hay un proceso, ni una metodología que apoye a los empleados para abordar de manera integral los problemas propios de las actividades de mantención y soporte a los clientes.
En el contexto actual se presentan fallas importantes en la provisión de recursos y en la coordinación de la cadena de atención a las peticiones de los clientes.
Estos problemas (no independientes unos de otros) son:

1. Los problemas derivados de “errores” en los productos actualmente en producción no son atendidos con la celeridad requerida.
2. No se cuenta con una coordinación adecuada para la atención de los requerimientos de software de clientes en producción.
3. No existe una metodología clara para abordar la gestión y asignación de recursos y prioridades para este tipo de tareas en el área de desarrollo, y más bien se hacen ad-hoc a cada caso.
4. No existe una infraestructura tecnológica adecuada que apoye el trabajo de soporte y mantención de los sistemas.
5. No hay una asignación clara de recursos de desarrollo a tareas relacionadas con la atención de los clientes de productos actualmente en producción.
6. Hay casos de reclamos de clientes que pueden tardar meses sin una justificación clara.
7. No hay una definición clara de la cadena de resolución de problemas, lo que produce confusión, y malos entendidos con los clientes.
8. La forma de trabajar usada actualmente ha permitido abrir un canal con el cliente, que brinda atención básica, pero que no es capaz de escalar en casos más complejos.
9. Algunos problemas denunciados por el cliente resultan ser sólo un error de responsabilidad del cliente. Sin embargo, tales problemas igual llegan al departamento de desarrollo, alargando aún más los tiempos de respuesta para los problemas de software que se deben resolver.
10. No siempre se hacen pruebas de las modificaciones hechas al software, lo que ha permitido introducir problemas en los productos en producción que pudieron haberse evitado.

Lo anterior ha resultado en una mala percepción del servicio prestado internamente por el departamento de desarrollo, y ha sido sindicado como una de las causas del deterioro del servicio prestado a los clientes finales.

Las consecuencias de no haber realizado antes este esfuerzo ya se están sintiendo, y van desde la pérdida de prestigio de la empresa hasta la fuga de clientes hacia la competencia.

Para enfrentar con éxito el desafío del crecimiento experimentado por la empresa, es necesario establecer una metodología de trabajo y formalizar los procesos correspondientes al ámbito de la mantención de software. Estos procesos deben integrarse a la cadena de valor de atención a los clientes de forma eficiente.

1.2 JUSTIFICACIÓN DE LA PROPUESTA

Dentro de los objetivos a mediano plazo de la empresa está el expandir su cuota de mercado y prestar servicios al extranjero. La forma actual del abordar el proceso de mantención no ha sido capaz de escalar para atender la creciente demanda, lo que es un riesgo para los objetivos de mediano plazo.

Se propone abordar este problema diseñando, e implementando el ciclo completo de solución de problemas del cliente. Esto supone la mejora y rediseño de los procesos actuales, integrando el proceso de gestión de problemas, y el de mantención de software. Este proceso se apoyará en TI de modo de agilizar la coordinación de las tareas, la transición entre las diferentes fases, y el personal a cargo de estas.
Se espera obtener como resultado procesos que permitan disminuir los tiempos de ciclo actual, y disminuir los tiempos de generación de indicadores para la gestión de los procesos involucrados.

1.3 OBJETIVO GENERAL

Mejorar la eficiencia en la gestión de problemas y mantención de software de la empresa Acepta.

1.4 OBJETIVOS ESPECÍFICOS

Para conseguir el objetivo general se definieron los siguientes objetivos específicos.

- Reducir tiempo de respuesta de un requerimiento de cliente, de 102 horas observadas, a un máximo de 7 horas para gestión de problemas, y 72 horas para mantención correctiva de software (promedio)
- Reducir el tiempo necesario para implementar cambios de proceso de 4 días a 24 horas
- Gestionar el 100% de los casos con apoyo de TI, versus el 45% actualmente

1.5 METODOLOGÍA

Para cumplir con los objetivos planteados, el desarrollo del trabajo se llevará a cabo en las siguientes fases:

- Levantamiento de los problemas actuales y su importancia, para lo cual se recurrirá a entrevistas y a registros históricos.
- Investigación bibliográfica para hacer un levantamiento del marco teórico y práctico en la industria para el tratamiento de la problemática presentada.
- Levantamiento de los procesos actuales, para lo cual se recurrirá a la documentación existente en el SGC (Sistema de gestión de calidad según norma ISO-9001).
- Diseño de los cambios y/o rediseño de procesos necesarios para satisfacer los objetivos de mejora planteados.
- Implementación de las mejoras de proceso y sus mecanismos de medición.
  Para la implementación de los procesos se propone el apoyo de estos a través de tecnología de workflows.
  Para la mejora continua se propone utilizar el proceso de implementación en fases definido para BPMS(Business Process Management System) basado en sistemas de información:

  - Definición del proceso
  - Implementación del proceso
  - Despliegue del proceso
  - Gestión del proceso
  - Medición del proceso

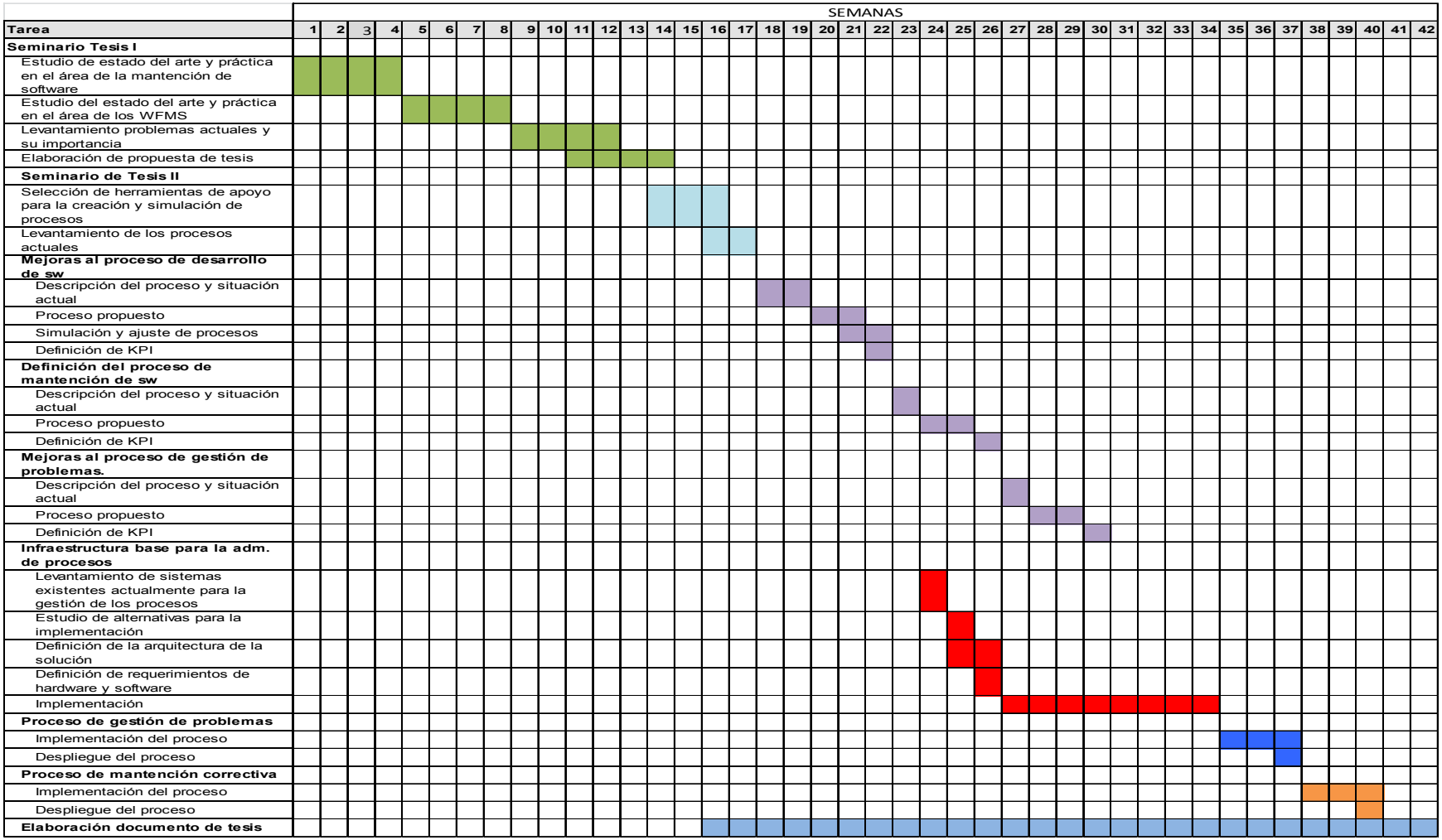
1.6 RESULTADOS ESPERADOS

Como resultado de esta tesis se espera obtener cambios en los procesos y prácticas actuales, que permitan, con su implementación, mejorar la eficiencia en la gestión de problemas. En particular se espera:

- Obtener el diseño de un proceso para la mantención de software, que permita afrontar de manera eficiente los cambios propios de la evolución del negocio de la empresa Acepta.
- Obtener un diseño del proceso de desarrollo de software mejorado para Acepta e integrado con el de proceso de mantención de software propuesto.
- Determinar e implementar prácticas que ayuden a mejorar la mantenibilidad de los productos de software de Acepta, y los tiempos asociados a cambios en el software.
- Obtener un diseño del proceso de gestión de problemas mejorado para Acepta e integrado con el proceso de mantención de software propuesto.
- Definición de los KPI de los procesos propuestos que permitan una gestión de mejoras a los procesos en base a criterios claros y establecidos.
- Definición de una infraestructura de TI que apoye la gestión de los procesos propuestos para Acepta, y que permita que esta gestión sea ágil.

1.7 PLAN DE TRABAJO

El plan de trabajo propuesto inicialmente para llevar a cabo esta iniciativa es el siguiente:



CAPÍTULO 2 MARCO CONCEPTUAL

2.1 ESTADO DEL ARTE Y DE LA PRÁCTICA EN EL ÁREA DE MANTENCIÓN DE SOFTWARE.

La mantención de software es reconocida como la etapa que sigue a la construcción y puesta en producción de un producto de software, y cuyos costos en sistemas de larga vida, superan el 60% del total [41]. En esta etapa se abordan los aspectos relacionados con la evolución del software para adaptarlo a los cambios requeridos por los clientes y su entorno.

El mantenimiento del software ha sido abordado en diferentes estándares. El estándar IEEE 1219 [22] lo define como “la modificación de un producto de software después de haber sido entregado con el fin de corregir defectos, mejorar el rendimiento u otros atributos, o adaptarlo a un cambio en el entorno”.

En el estándar ISO 12207 [25], Procesos del ciclo de vida del software, se define que “el proceso de mantenimiento contiene actividades y tareas realizadas por el mantenedor. Este proceso se activa cuando el producto de software sufre modificaciones en el código y la documentación asociada, debido a un problema o a la necesidad de mejora o adaptación. El objetivo es modificar el producto de software existente preservando su integridad. Este proceso incluye la migración y retirada del producto de software. El proceso termina con la retirada del producto de software”. El mantenedor es la organización que presta el servicio de mantenimiento.

En el libro de Ingeniería de software de Pressman [37], se dice que, “La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, y a los cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente”.

En el estándar ISO12207 [25] además se define como “mantenedor” a la organización que desempeña las actividades de mantención. En el SWEBOK [21] se utiliza también para referirse a los individuos que desempeñan estas actividades.

El SWEBOK [21] dice que el contacto con los desarrolladores y participación temprana de los mantenedores reduce el esfuerzo de mantención. Esto también es resaltado en el ISO/IEC 14764 [24] (sección consideraciones para la ejecución).

Se reconocen cinco tipos de Mantenimiento de Software dependiendo de los requerimientos de los usuarios:

- Mantenimiento Adaptativo
- Mantenimiento Correctivo
- Mantenimiento de Emergencia
- Mantenimiento Perfectivo
- Mantenimiento Preventivo

Mantenimiento Adaptativo: Es la modificación de un producto de software, una vez puesto en producción, y cuyo objetivo es adaptar el producto a cambios del entorno (hardware, software, requerimientos), de modo de mantenerlo operativo.

Mantenimiento Correctivo: Es la modificación de un producto de software, una vez puesto en producción, y cuyo objetivo es localizar y corregir los defectos del producto causantes de fallas del sistema en producción.

Las fallas son la diferencia entre el comportamiento del sistema real y el esperado según la especificación del sistema.
Los defectos en general se localizan en las fases de requerimientos y codificación del ciclo de vida.

Mantención de emergencia: Es la modificación no programada de un producto de software, una vez puesto en producción, y cuyo objetivo es mantener el sistema operativo a la espera de una mantención correctiva.

Mantenimiento Perfectivo: Es la modificación de un producto de software, una vez puesto en producción, y cuyo objetivo es mejorar su rendimiento o mantenibilidad.

Mantenimiento Preventivo: Es la modificación de un producto de software, una vez puesto en producción, y cuyo objetivo localizar y corregir los defectos del producto que potencialmente causarían fallas en producción.

En el estándar ISO/IEC 14764 [24] se describe en detalle la gestión del proceso de mantención de software descrito en ISO12207 [25]. Se establecen los tipos de mantención anteriormente descritos, guías para la planificación, ejecución, control, revisión, evaluación, y cierre del proceso de mantención de software.
El ISO/IEC 14764 [24] se define como un estándar aplicable a organizaciones de mantención tanto internas como externas, sin importar el modelo de ciclo de vida y el tamaño del software a mantener, no siendo aplicable a software de vida corta o soluciones temporales.

	Corrección	Mejora
Proactiva	Preventiva	Perfectiva
Reactiva	Correctiva	Adaptativa

Tabla 1: Categorías de mantención de software

En la tabla 1 se muestran las categorías de mantención de software establecidas en ISO/IEC 14764 [24].

Respecto del esfuerzo de mantención, el SWEBOK [21] menciona que, sobre el 80% de este corresponde a acciones no correctivas, y que una mala práctica de agrupar juntas las correcciones y las mejoras por parte de los gerentes ha contribuido a difundir una mala percepción sobre los altos costos de las actividades de corrección.

El SWEBOK [21] define cuatro grupos de problemas clave que se deben resolver para asegurar una efectiva mantención del software, y que se mencionan a continuación.

2.1.1 Problemas técnicos

Entendimiento limitado: Se refiere a qué tan rápido un ingeniero de software puede entender donde hacer un cambio o corrección en software no desarrollado por él. Se dice que entre el 40% y 60% del esfuerzo de mantención se ubica en este ítem.

Testing: Se refiere al costo significativo, en términos de dinero y tiempo, que puede resultar de repetir continuamente un test completo a una aplicación grande.

Análisis de impacto: Se refiere a cómo llevar, a costos razonables, el análisis completo del impacto de un cambio en el software existente. El mantenedor debe tener un conocimiento profundo del contenido y estructura del software, para poder llevar a cabo dicha actividad. Como resultado se debe obtener todos los sistemas y productos de software afectados por el requerimiento de cambio, una estimación de los recursos necesarios parta llevar a cabo dichos cambios, y los riesgos asociados a efectuar los cambios.

Mantenibilidad: El IEEE [23] define manteniblidad como la facilidad con la que el software puede ser mantenido, mejorado, adaptado o corregido para satisfacer un requerimiento especificado. También es definida como una de las características de calidad del software en ISO 9126 [26]. SWEBOK [21] menciona que la poca importancia dada a esta característica durante el proceso de desarrollo suele causar la mayor parte de las dificultades en la posterior comprensión de los programas y análisis de impacto.

2.1.2 Problemas de administración

Alineamiento con los objetivos organizacionales: La mantención de software frecuentemente tiene por objetivo alargar la vida del software tanto como sea posible. Además, esta puede estar impulsada por la necesidad de satisfacer las demandas de los usuarios por actualizaciones y mejoras del software. En ambos casos el retorno de inversión es mucho menos claro que en el caso del desarrollo, y la visión a nivel de la alta administración es de una actividad que consume muchos recursos sin beneficios cuantificables para la organización.

Personal: Se refiere a cómo atraer y mantener al personal de mantención. La mantención frecuentemente no es vista como un trabajo glamoroso. El personal de mantención es frecuentemente visto como "ciudadanos de segunda clase" y la moral por lo tanto sufre.

Procesos: Los procesos de software son un conjunto de actividades, métodos, prácticas y transformaciones que las personas usan para desarrollar o mantener software y sus productos asociados. El mantenimiento comparte actividades con el proceso de desarrollo, y además requiere de actividades propias de la mantención, y que son un desafío para la administración.

Aspectos organizacionales: Describe qué organizaciones o funciones serán responsables por la mantención del software. Aquí lo importante es asignar la responsabilidad de mantención en un solo grupo o persona, no importando la estructura de la organización.

Outsourcing: Es la opción utilizada por grandes corporaciones para el software que no es de misión crítica y no es parte del "core" de su negocio. Uno de los mayores desafíos en este caso es determinar el alcance del servicio de mantención y los detalles contractuales de este.

Estimación del costo de mantención: Según ISO/IEC 14764 las dos formas más populares para la estimación de recursos para mantención son el uso de modelos paramétricos y el uso de experiencia.

Modelos paramétricos: En estos modelos es importante el uso de datos capturados de proyectos pasados. Ejemplos de estos modelos son PRICE S [6], SLIM [5], y COCOMO [4].

Experiencia: En la forma de juicio experto (por ej. usando la técnica Delphi), analogías, y estructuras para la división del trabajo (WBS), son varias aproximaciones que deben ser utilizadas para aumentar los datos obtenidos de los modelos paramétricos. La mejor aproximación a la estimación de mantención es la combinación de datos empíricos y de experiencia.

2.1.3 Medición en la mantención de software

Medidas específicas: IEEE 1219 [22], ISO 9126 [26] sugieren medidas específicas para programas de medición en la mantención de software.

Esta lista incluye métricas para cada una de las cuatro subcaracterísticas de "mantenibilidad".

Métricas internas (del producto)

- **Analizabilidad**: Métricas que ayudan a predecir el esfuerzo de los mantenedores o recursos utilizados en intentar diagnosticar las deficiencias, o en identificar las partes a ser modificadas de un sistema.
    - Registro de actividad: Es una medida de qué tan exhaustivamente es registrado el estado del sistema.
    - Preparación de la función de diagnóstico: Es una medida de qué tan completa es la provisión de funciones de diagnóstico, entendiendo por función de diagnóstico, aquella que analiza una falla y provee una salida para el usuario o log, con una explicación de la falla.

- **Cambiabilidad**: Métricas que ayudan a predecir del esfuerzo de los mantenedores asociado con la implementación de una modificación especificada.
    - Registrabilidad de cambios: Es una medida de qué tan completa es la documentación de los cambios a las especificaciones y módulos de programas.

- **Estabilidad**: Métricas que ayudan a predecir qué tan estable será el sistema después de una modificación.
    - Impacto del cambio: Es una medida de la frecuencia en que se observan reacciones adversas después de modificaciones al sistema.
    - Localización del impacto de la modificación: Es una medida de qué tan grande es el impacto de una modificación sobre el sistema.
- **Facilidad de prueba**: Métricas que ayudan a predecir la cantidad de ayuda que el sistema tiene predefinidas para propósitos de prueba.
    - Integridad de función de prueba predefinida: Es una medida de qué tan completa es la capacidad de las funciones de prueba predefinidas.
    - Autonomía de facilidad de prueba: Es una medida de qué tan independientemente se puede probar el sistema de software.
    - Observabilidad del progreso de las pruebas: Es una medida de qué tan completamente los resultados de las pruebas predefinidas, pueden ser desplegadas durante las pruebas.

Conformidad de mantenibilidad: Es una medida de qué tan conforme está el sistema de acuerdo a regulaciones, estándares y convenciones aplicables.

Métricas externas (del entorno)

- **Analizabilidad**: Métricas para medir el esfuerzo necesario mientras se trata de diagnosticar la causa de las fallas o identificar los ítems a ser modificados.
    - Capacidad de auditoría: Es una medida de qué tan fácil es para un usuario (o mantenedor) identificar la operación específica que causa una falla.
    - Soporte de la función de diagnóstico: Es una medida de qué tan capaces son las funciones de diagnóstico para apoyar los análisis de causa.
    - Capacidad de análisis de falla: Es una medida de la habilidad de los usuarios o mantenedores para identificar la operación específica que causa la falla.
    - Eficiencia del análisis de falla: Es una medida de cuán eficientemente un usuario o un mantenedor puede analizar la causa de una falla.
    - Capacidad para monitorear estado: Es una medida de qué tan fácil es obtener data monitoreada para las operaciones que causan fallas durante la ejecución actual del sistema.

- **Cambiabilidad**: Métricas que ayudan a medir el esfuerzo necesario cuando se tratan de implementar los cambios al sistema.
    - Eficiencia del ciclo de cambio: Es una medida de cuán probable es que el problema de un usuario pueda ser resuelto en un período de tiempo aceptable.
    - Tiempo transcurrido en la implementación de un cambio: Es una medida de qué tan fácilmente un mantenedor puede cambiar un software para resolver una falla.
    - Complejidad de modificación: Es una medida de cuán fácilmente un mantenedor puede cambiar un software para resolver un problema.
    - Modificabilidad parametrizada: Es una medida de cuán fácilmente un usuario o mantenedor puede resolver un problema de software solamente cambiando un parámetro.
    - Capacidad de controlar el cambio del software: Es una medida de qué tan fácilmente un usuario puede identificar una versión revisada del software.

- **Estabilidad**: Métricas usadas para medir el comportamiento inesperado del sistema después de ser modificado.
    - Radio de cambio exitoso: Es una medida de qué tan bien puede el usuario operar el software después de una mantención sin nuevas fallas.

- **Facilidad de prueba**: Métricas para medir el esfuerzo requerido para probar el sistema una vez modificado.
    - Disponibilidad de funciones de prueba predefinidas: Es una medida de cuán fácilmente un usuario o mantenedor puede ejecutar pruebas operacionales sobre el sistema sin la preparación de facilidades de prueba adicionales

- Eficiencia de pruebas sucesivas: Es una medida de qué tan fácilmente un usuario o mantenedor puede ejecutar pruebas operacionales y determinar cuando un software esta listo para ser liberado.
- Capacidad de reiniciar las pruebas: Es una medida de cuán fácilmente un usuario o mantenedor puede ejecutar pruebas operacionales con puntos de control después de mantención.

Conformidad de mantenibilidad: Es una medida de qué tan cercana es la adherencia del sistema a varios estándares, convenciones y regulaciones.

Muchas de las métricas establecidas en ISO 9126 [26] requieren un alto grado de madurez de la organización [17], por lo que cada organización debe decidir cuál es apropiada para su realidad.

2.1.4 El proceso de mantención de software

El proceso de mantenimiento de software es uno de los procesos principales dentro de lo que se define como los procesos del ciclo de vida del software (ISO/IEC 12207 [25]).

El proceso de mantención de software define las actividades a realizar y el detalle de las entradas y salidas de cada una de ellas.

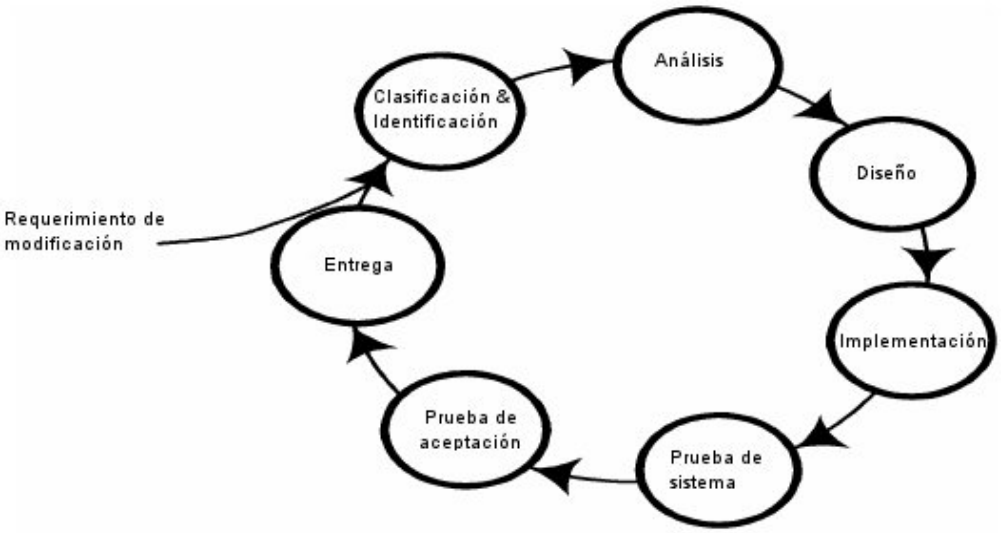


Figura 1: Proceso de mantención de software según IEEE 1219

En la figura 1, se muestra el proceso de mantención de software sugerido por el estándar IEEE 1219 [22].

En este caso el proceso está compuesto de siete fases, y se caracteriza por ser iterativo y en cascada.

El proceso se inicia con un requerimiento de modificación, término que incluye informes de problemas y errores, o cambios.

Clasificación e Identificación: En esta fase se identifica, clasifica, se asigna una prioridad inicial, y se realiza una estimación de esfuerzo inicial del requerimiento de modificación. Además se realiza un análisis para aceptar o rechazar el requerimiento.

Análisis: En esta fase se estudia la viabilidad y alcance del requerimiento de modificación anteriormente clasificado y priorizado. Además se prepara un plan preliminar de diseño, implementación, pruebas y entrega del software. Las salidas típicas de esta fase son:

- Informe de viabilidad del requerimiento de modificación recibido.
- Informe de análisis detallado
- Actualización de requisitos
- Lista de modificaciones preliminares
- Estrategia de pruebas
- Plan de implementación.

Diseño: En base a los documentos asociados al proyecto de software a modificar y a la salida de la actividad de análisis, se realiza el diseño de las modificaciones requeridas. En esta fase se identifican los módulos a modificar, se actualiza la documentación asociada (diagramas de flujo, etc), y se crean casos de prueba que correspondan. Las salidas de esta fase son:

- Lista de modificaciones revisada
- Guía básica del diseño actualizada
- Planes de prueba actualizados
- Análisis de prueba actualizados
- Requisitos verificados
- Plan de implementación verificados
- Lista de restricciones y riesgos bien verificados

Implementación: En base a los documentos asociados al proyecto a modificar y a la salida de la actividad de diseño se realiza la implementación de las modificaciones requeridas. En esta fase se realizará de manera iterativa e incremental, hasta obtener el entregable, las siguientes actividades:

- Codificación y pruebas unitarias
- Integración
- Análisis de riesgo e integración
- Revisión de pruebas de sistema

Finalmente como salida de esta fase se obtiene:

- Software actualizado
- Documentación de diseño actualizada
- Documentación de prueba actualizada
- Documentación de usuario actualizada
- Material de entrenamiento actualizado
- Informe de revisión de pruebas de sistema

Pruebas de sistema: Estas pruebas se realizan sobre el sistema modificado. Esta actividad debe ser realizada por una organización independiente del desarrollo. Las pruebas de regresión son parte de las actividades de esta fase de modo de asegurarse de no introducir fallas que no existieran antes de la actividad de mantención.

Las actividades realizadas en esta fase son:

- Pruebas funcionales del sistema.
- Pruebas de interfaces
- Pruebas de regresión
- Revisión de pruebas de aceptación

Finalmente como salida de esta fase se obtiene:

- Sistema probado e integrado completamente.
- Informe de pruebas
- Informe de revisión de pruebas de aceptación

Pruebas de aceptación: Estas pruebas se realizan sobre el sistema modificado y completamente integrado. Esta actividad debe ser realizada por el cliente, un usuario del sistema o un tercero designado por el cliente, y el software probado debe estar bajo control de configuración.

Las actividades realizadas en esta fase son:

- Ejecutar pruebas de aceptación a nivel funcional
- Ejecutar pruebas de interoperabilidad
- Ejecutar pruebas de regresión

Finalmente como salida de esta fase se obtiene:

- Nueva línea base del sistema
- Informe de pruebas de aceptación

Entrega: Para la liberación de la nueva versión del software, se establece en esta fase las siguientes actividades:

- Conducir una auditoría de configuración física
- Notificar a la comunidad de usuarios
- Desarrollar una versión de archivo del sistema para backup
- Ejecutar la instalación y entrenamiento.

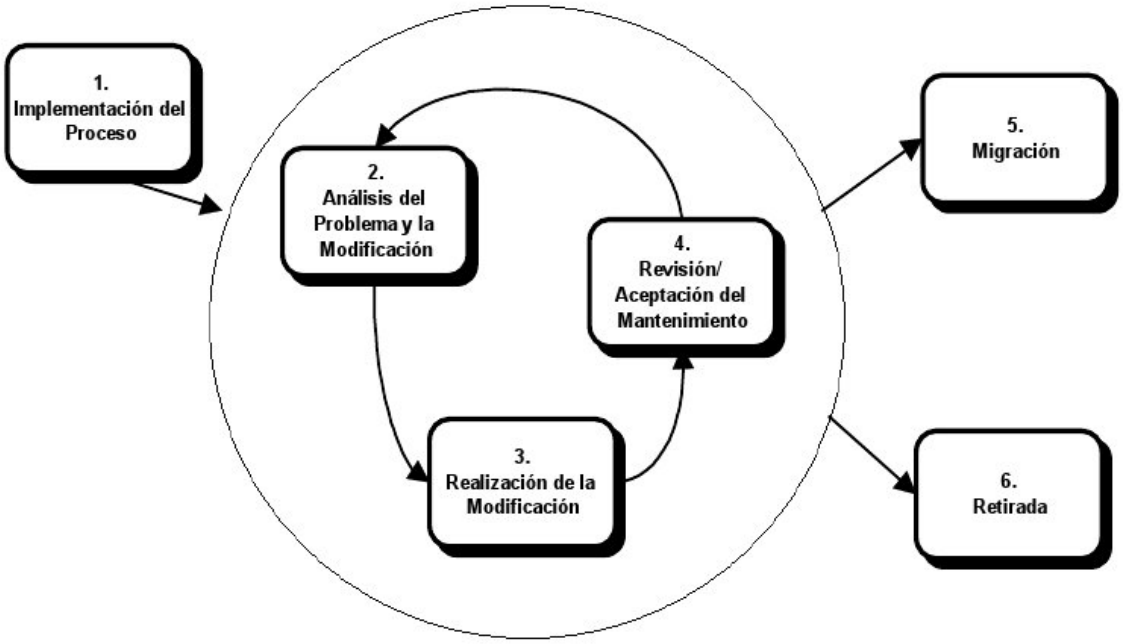


Figura 2: Proceso de mantención de software según ISO/IEC 14764

En la figura 2 se muestra el proceso de mantención de software sugerido por el estándar ISO/IEC 14764 [24].

En este caso el proceso se compone de seis actividades, las que se inician con la implementación de proceso, para luego entrar en un ciclo de mejora continua el que es solo interrumpido para efectuar la retirada del producto de software o en el evento de una migración.

Implementación del proceso: Durante esta actividad se desarrollan los planes y procedimientos que se ejecutarán durante el proceso de mantención. El plan de mantenimiento se debería desarrollar en paralelo al plan de desarrollo. También se deben establecer las interfaces organizacionales necesarias, entendiendo por interfaces las formas de comunicación con la organización involucrada.

Análisis del problema y la modificación: Esta actividad se activa después de la transición, y es ejecutada iterativamente en la medida que las necesidades de modificación lo requieren.

Realización de la modificación: Durante esta actividad el mantenedor desarrolla y prueba la modificación del producto de software.

Revisión/Aceptación del mantenimiento: Esta actividad asegura que las modificaciones realizadas al producto de software son correctas y que estas se llevaron a cabo de acuerdo a los estándares aprobados y usando la metodología correcta.

Migración: Esta actividad cubre la necesidad de modificar un sistema para ser ejecutado en un ambiente diferente. Durante su ejecución se determinan las acciones necesarias para llevar a cabo la migración, y se desarrollan y documentan los pasos requeridos para la migración.

Retirada: Una vez que el software ha alcanzado el fin de su vida útil, este debe ser retirado. Un análisis debe ser llevado para asistir la decisión de retirar el producto de software. El análisis es frecuentemente económico y puede estar incluido en el plan de retirada.

2.1.5 Modelos de madurez en el área de mantenimiento de software

Los modelos de madurez son usados para la evaluación de los procesos de una organización. Esta evaluación se hace en relación a las mejores prácticas del área de conocimientos que abarcan, y establece niveles desde un nivel inicial a un nivel óptimo, esto bajo la premisa que a mejores procesos, mejores productos.
En el área de mantenimiento de software se conocen los siguientes modelos de madurez:

- CMMi-DEV [40]: Modelo de madurez y capacidades creado por el SEI, es un modelo de madurez para la mejora continua de los procesos de software, y cubre el ciclo de vida del software desde su concepción hasta la entrega y mantención del software. Este modelo tiene como propósito ayudar a las organizaciones a mejorar sus procesos de desarrollo y mantención, tanto de productos como de servicios.
- SM3 (Software maintenance maturity model) [13]: Modelo de madurez centrado en el proceso de mantención de software desarrollado por April, Hayes, Abran, y Dumke , se basa en una estructura similar a la de CMMi y está diseñado para ser un complemento de este. La arquitectura de este modelo se basa en la utilizada en CMMi y agrega a ésta dos elementos: Una categoría "*roadmap*" para definir con mayor precisión las KPAs, y referencias detalladas a *papers* y ejemplos sobre cómo implementar las prácticas.
- CM3(Corrective maintenance maturity model) [31][30]: Modelo de madurez especializado en mantención correctiva creado por Mira Kajko. Es parte del EM3(Evolution and Maintenance Management Model) actualmente en desarrollo por el SYSLAB de la Universidad de Estocolmo.

2.1.6 Estado de la práctica en el área de mantención de software

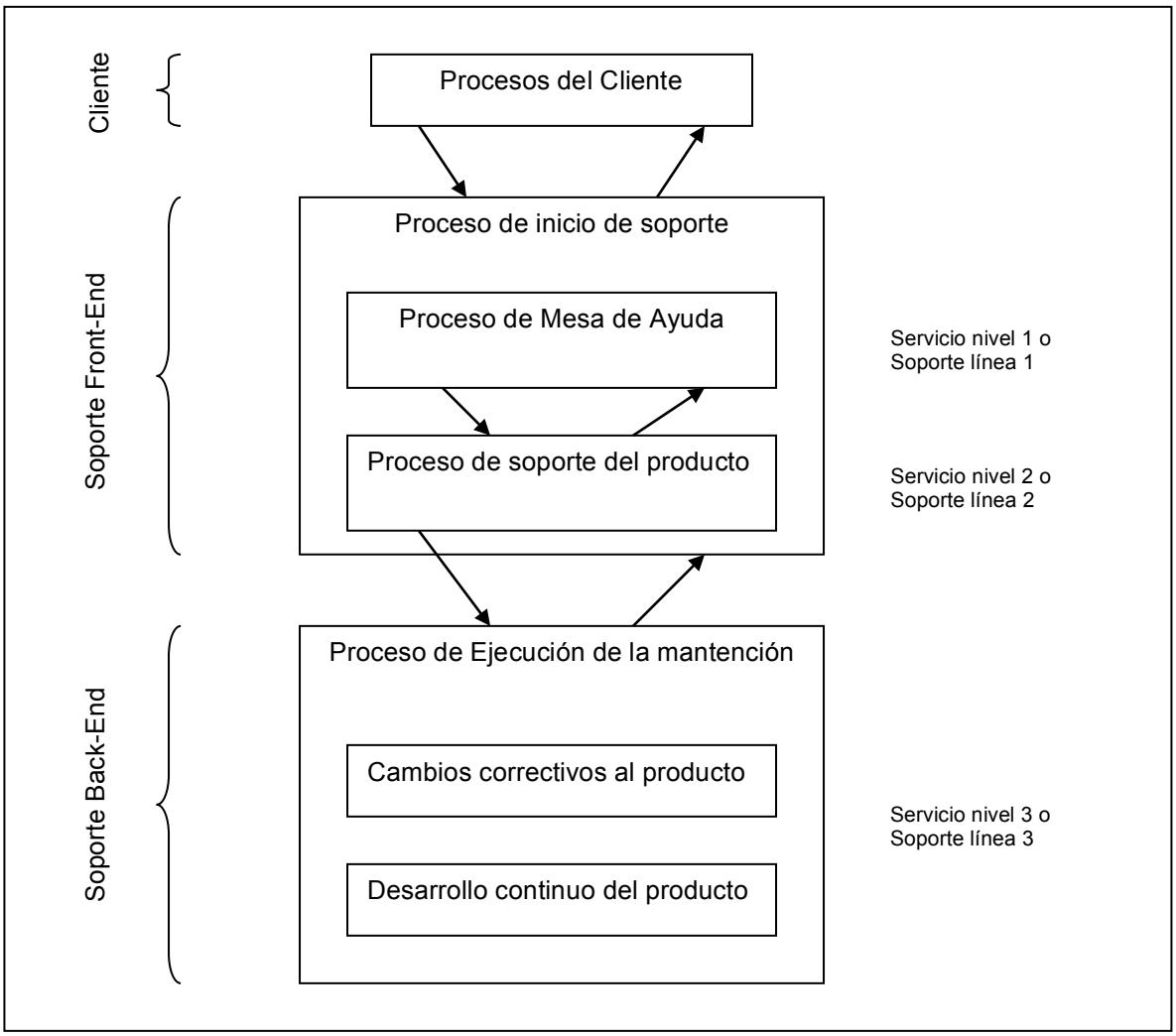


Figura 3: Organización de mantención

Una vez entregado el software a producción, lo común en las empresas es que comiencen a operar con el cliente a través de estructuras organizacionales como las de la figura 3. Esta estructura organizacional ordena la función de soporte al cliente, siendo parte de los servicios post-venta de los productos y siendo parte integral de los servicios ofrecidos/contratados en la actualidad.

Estas organizaciones pueden pertenecer a la misma empresa o ser parte de otras empresas en las cuales se ha externalizado la función. A esto, Kajko le llama Virtual IT Enterprise, y puede tener diferentes configuraciones según la empresa [31], pero la relación básica permanece entre las partes.

El soporte Front-end recibe los reportes de problemas de software de parte del cliente, ayuda al cliente en la operación de los productos, filtra los problemas “reales” y los transfiere al soporte de back-end, y entrega la solución de los problemas desde el soporte back-end al cliente.

El soporte Front-end gestiona la comunicación con el cliente, hace seguimiento a los casos y da feedback continuo a los clientes, mientras el soporte back-end se concentra en la solución de los problemas de software reales que afectan la operación del cliente.

La gestión de problemas, en este tipo de organizaciones, es realizada mediante técnicas de administración de colas, en contraposición a las técnicas de administración de proyectos utilizadas durante el proceso de desarrollo del producto de software [13].

La gestión de otros tipos de cambios como son las de tipo perfectivo, adaptativo, y preventivo, y dependiendo de su magnitud, usan el enfoque de administración de proyectos.

La función técnica de la mantención de software puede ser llevada por la misma organización que desarrolló el producto de software como por una totalmente distinta, y son enfoques que dependen de cada organización [36].

Desde el punto de vista de los procesos, la mantención de software comparte muchos de los procesos y tareas del proceso de desarrollo de software, por lo que se reutilizan varios procesos utilizados durante el desarrollo, incorporándolos en el proceso, y adaptándolos para la evolución del producto de software.

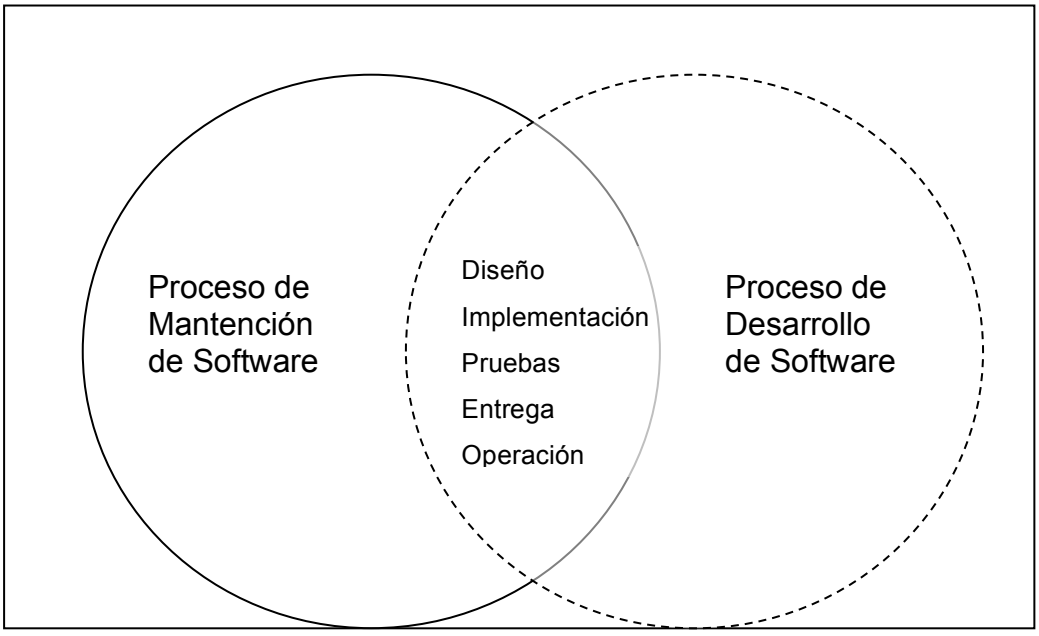


Figura 4: Intersección entre el proceso de desarrollo y el de mantención de SW

Cada pieza de software a mantener tiene diferentes desafíos y exigencias técnicas y de negocio. En estos casos, es de uso común buscar experiencias similares en la forma de patrones, ya través de su conocimiento ajustar las estrategias a seguir. Un conjunto de patrones útil para consultar al momento de abordar la mantención de software se puede consultar en el libro Object-Oriented Reegineering Patterns [44].

2.1.7 Test Driven Development (TDD)

Desde el mundo de las metodologías ágiles es importante destacar las prácticas de TDD, y el aporte que pueden ser para el mantenimiento del software. En simple, TDD es la implementación de pruebas automatizadas que den garantía de la correctitud del código que prueban. Donde, primeramente se desarrollan los tests, para luego implementar el código que los satisfaga, agregando nuevos tests y refactorizando el código probado de manera incremental. [43].

La importancia de TDD y la automatización de pruebas para la mantención de software, viene dada por sus beneficios en el traspaso de conocimiento. Lo anterior viene dado a partir de la especificación del comportamiento que debe tener el software, y que se encuentra directamente en el código de pruebas. El código generado tiende a ser más modular, y su set de prueba automatizado permite detectar rápidamente la introducción de errores al modificar el código durante la etapa de mantención.

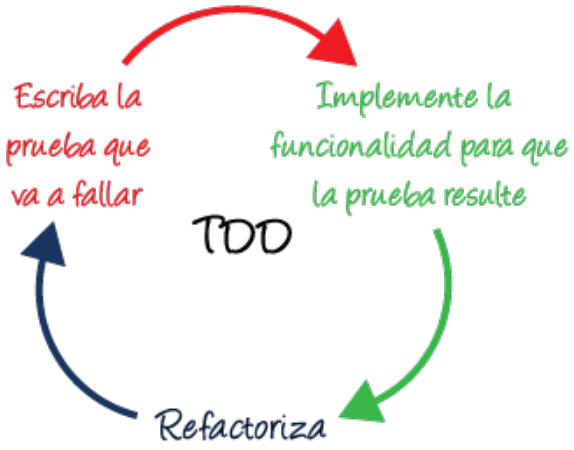


Figura 5: Proceso de TDD

2.2 ESTADO DEL ARTE Y DE LA PRÁCTICA EN EL ÁREA DE WFMS.

Un Sistema de administración de flujos de trabajo o WfMS por sus siglas en inglés, apoya los procesos de negocio, asegurando que la información correcta es entregada a la persona correcta en el tiempo que corresponde, o es entregada al sistema computacional en el momento correcto. Un WfMS no realiza ninguna de las tareas en el proceso [9].

Según la Workflow Management Coalition(WfMC, organización internacional responsable por la estandarización de los WfMS), un workflow se ocupa de la automatización de procedimientos donde documentos, información o tareas son pasadas entre participantes de acuerdo a un conjunto de reglas definidas para lograr, o contribuir a, un objetivo de negocio general. Un WfMS es así un sistema que define, administra y ejecuta completamente workflows a través de la ejecución de software cuyo orden de ejecución es dirigida por una representación computacional de lógica de workflow.

La importancia de los WfMS está dada por la adopción en las organizaciones de la gestión dirigida por procesos. Una vez que la organización ha identificado los procesos críticos, los WfMS se transforman en una herramienta muy útil a la hora de mejorar tiempos respuesta, mediante la automatización del flujo de información y tareas a través de los participantes del proceso.

Las aplicaciones de workflow existieron mucho antes que los WfMS. En un inicio estas se implementaron como aplicaciones que automatizaban el flujo de información y tareas, y cuyo desarrollo estaba más bien basado en un repositorio compartido de datos. Los WfMS nacen como una propuesta del mercado de software para la automatización de workflows, sin embargo al no existir un estándar, no proveen unicidad de conceptos, ni interoperabilidad en las soluciones.

En 1995 se publica el modelo de referencia para WfMS, que pretende llenar el vacío de estandarización existente, y sentar las bases para la interoperabilidad de estos productos basado en estándares para cada una de las funciones que se pueden realizar [7].

La administración de workflows tiene por objetivo el modelamiento y control de la ejecución de procesos de aplicación dinámicos, y en ambientes, técnica y organizacionalmente, heterogéneos [2].

En la actualidad los conceptos de workflow han evolucionado más allá de la automatización del flujo de tareas entre personas, y sientan las bases para la integración de sistemas en la automatización y gestión de procesos de negocio de las empresas en lo que se ha denominado BPMS (Business Process Management System).

2.2.1 Conceptos

Algunos conceptos básicos en esta área son:

Workflow: Automatización o facilitación computarizada de un proceso, completa o parcialmente.

Modelo de workflow o definición de proceso: Es la representación computacional de un proceso. Este define las condiciones de inicio y fin del proceso, las actividades en el proceso, flujos de control (o reglas de navegación) y flujos de datos sobre estas actividades. Los modelos implementados en diferentes WfMS son un poco diferentes. Los primeros sistemas implementaban la lógica de workflow a nivel de código fuente, lo que hace muy difícil la modificación de la lógica. Muchos otros modelos, tales como grafos dirigidos, grafos dirigidos condicionales, redes de Petri, modelos orientados a objeto, se han desarrollado con posterioridad.

Actividad: Es un paso lógico en un workflow. Incluye información sobre las condiciones de inicio y fin, los usuarios que participan, las herramientas y/o datos necesarios para completar la actividad, y las restricciones sobre cómo la actividad debe ser completada.

Instancia de proceso: Es la ejecución de una definición de proceso conducida por el WfMS. El WfMS interpreta la definición de proceso y controla la instanciación de procesos y actividades, agregando ítems de trabajo a la “lista de trabajos” del usuario e invocando herramientas de aplicación como sea necesario. Los datos manipulados por las herramientas de aplicación son referenciados como datos de aplicación de workflow, algunos de los cuales son también usados para controlar la ejecución del workflow en conjunto con la definición de proceso. Esta parte de los datos de aplicación es llamada datos relevantes de workflow.

Lista de trabajo: Lista de tareas de una instancia de proceso manejada por el WfMS para un usuario en particular. Esta lista de tareas es expuesta al usuario a través de una aplicación cliente.

2.2.2 Topologías de workflow

De acuerdo a las características de workflow, los métodos de modelamiento, las tecnologías subyacentes, y los modos de ejecución, los productos WfMS existentes pueden ser categorizados de la siguiente manera [38]:

Estructurado o Ad-hoc: Para un workflow estructurado, toda la información necesaria para definir el proceso de negocio puede ser obtenida a través del análisis y modelamiento de proceso antes de su ejecución real. El proceso se repite una y otra vez en el mundo real. Cuando la definición del proceso es terminada, esta nunca o rara vez cambia. Un ejemplo de esto es el procesamiento de formularios, donde un conjunto de formularios son llenados y enrutados a través de una serie de pasos. Por otra parte un workflow Ad-hoc tiene menos repetitividad, y algunos de los parámetros necesarios para definir el proceso pueden ser imposibles de ser determinados por adelantado y son especificados en tiempo de ejecución. Algunas excepciones al proceso normal pueden también suceder. Este carácter dinámico causa muchas dificultades a los WfMSs en el modelamiento y en la ejecución.

Centrado en el documento o centrado en el proceso: Los WfMS centrados en el documento tienen por objetivo enrutar documentos electrónicos y/o imágenes a través de múltiples personas para su revisión o procesamiento. Los WfMS centrados en el proceso, tratan de modelar el proceso de negocios como una serie de pasos interdependientes. En cada paso, hay algunos objetos de datos a ser procesados ya sea la invocación de las herramientas (aplicaciones) apropiadas, por los mismos usuarios, o por el WfMS. Estos objetos de datos son usados para construir los objetos de datos a ser pasados a otros pasos.

Basados en correo electrónico o basados en bases de datos: Los WfMS basados en correo electrónico usan sistemas de e-mail en el paso de mensajes, distribución de datos y notificación de eventos mientras se ejecuta la instancia de proceso. Este método es ampliamente adoptado por sistemas de gama baja y se ejecutan de manera poco acoplada. Por otro lado los WfMS basados en bases de datos almacenan todos los datos (incluyendo datos de aplicación) necesarios en algún tipo de sistema administrador de base de datos. Una instancia de ejecución es el proceso de recuperar y procesar estos datos.

Task-pushed o Goal-pulled: Los WfMS pertenecientes al primer tipo ejecutan las actividades del proceso, una por una. Cuando una actividad es terminada, las siguientes actividades son creadas y activadas. Después que todas las actividades han terminado, el proceso se considera terminado. Este modo de ejecución es implementado por la mayoría de los WfMS centrados en el proceso. En un WfMS “goal-pulled”, un proceso es considerado como un objetivo. El objetivo es dividido primero en múltiples pasos ejecutables e interdependientes. Cada uno de estos pasos puede ser visto como sub-objetivos que pueden ser divididos nuevamente. Cuando todos los sub-objetivos divididos han sido finalizados, la instancia se ejecuta hasta el punto de fin. Este modo de ejecución se prevé para próximas generaciones de WfMS [10,38].

2.2.3 El modelo de referencia WfMC

La Workflow Management Coalition es un grupo de empresas que se unieron en un esfuerzo conjunto por identificar las áreas funcionales, y las especificaciones de desarrollo apropiadas para la implementación de productos de workflow que permitan la interoperabilidad entre diferentes productos, y la integración de los productos de workflow con otros servicios de TI, como correo electrónico y administración de documentos. Como resultado de este esfuerzo se creó un modelo de referencia para los WfMSs, identificando sus características, terminología y componentes [7].

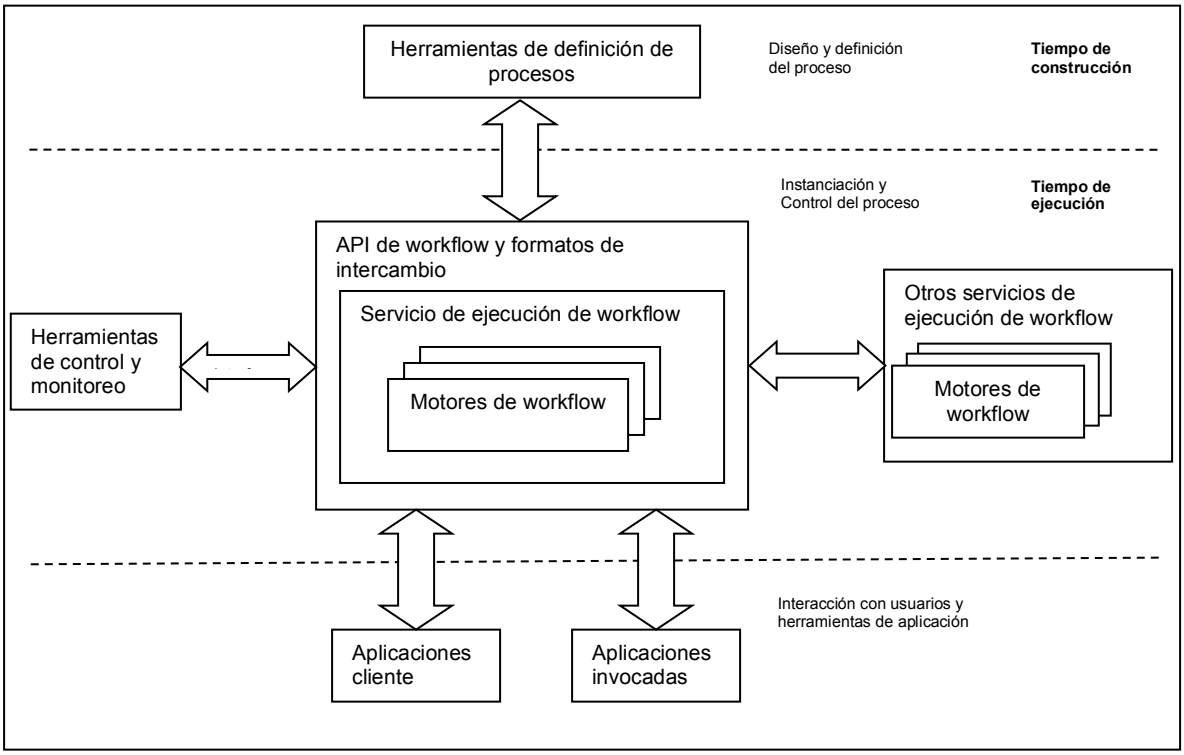


Figura 6: Modelo de referencia WfMC

El modelo de referencia sólo define un framework de acuerdo al cual pueden ser implementados los WfMS. No se define ni consideran detalles técnicos, los cuales son dejados a los investigadores y desarrolladores.

Herramientas de definición de procesos: Son usadas para describir los procesos de manera tal que puedan ser procesados e interpretados por un motor de workflow.

Aplicaciones clientes: Son usadas por los usuarios para interactuar con el workflow. La interacción se basa principalmente en el concepto de listas de trabajo (worklist). Además el usuario puede hacer uso de otras funciones como la petición de inicio de un tipo particular de proceso, consultar los ítems de trabajo encolados para cierto participante, etc.

Aplicaciones invocadas: Representan aplicaciones de software ya existentes que un WfMS puede utilizar para la realización de algunas actividades

Motores de workflow: Servicio de software o “motor” que provee un ambiente para la ejecución de una instancia de proceso. Interpreta la definición de procesos, permite el control de instancias de proceso (creación, activación, suspensión, término, etc), y navegación entre actividades.

Herramientas de control y monitoreo: Usadas para controlar y monitorear diferentes aspectos del workflow como la administración de usuarios, administración de roles y su asignación a los usuarios, operaciones de auditoría, funciones de supervisión de procesos, etc.

2.2.4 Procesos y metodologías de desarrollo para workflow

Al investigar sobre este tema, no se encontraron referencias fuertes sobre metodologías para este tipo de desarrollo. Se encontraron papers de diferentes autores que dan cuenta de una inquietud en el mundo académico por la falta de estudios sobre las propiedades específicas de los procesos de desarrollo de aplicaciones de workflow, en comparación con los procesos de desarrollo de software ampliamente estudiados [1,15,18,14,9].

2.2.5 Business Process Management (BPM) y Business Process Management Systems (BPMS)

BPM es un conjunto de herramientas para modelar, gestionar y optimizar los procesos de negocio de la organización, uniendo personas, conocimiento, y sistemas de negocio. Esta solución ofrece una detallada y actualizada vista panorámica sobre la organización, que mejora la toma de decisiones, la planificación del escenario, y la gestión de la organización.

Según Smith y Fingar [39], BPM toma prestados y combina características de diferentes herramientas y tecnologías, pero difiere en su foco central sobre los procesos. Los procesos pasan a ser ciudadanos de primera categoría tanto en la parte administrativa como en la parte tecnológica.

BPM se aleja del cambio total propuesto por la reingeniería y se acerca a las teorías de mejoramiento continuo de los procesos para mejorar los servicios brindados a los clientes. El mejoramiento continúo está basado fuertemente en la tecnología que soporta los procesos, haciendo del cambio de procesos una tarea dinámica, lo que permite materializar los cambios rápidamente y reaccionar a los cambios del entorno.

“BPM reconoce que el cambio es tan fundamental para el negocio como la ley de gravedad a la física, y que la agilidad es por lo tanto un requerimiento fundamental de la arquitectura empresarial” [39]

2.2.6 El ciclo de vida de BPM

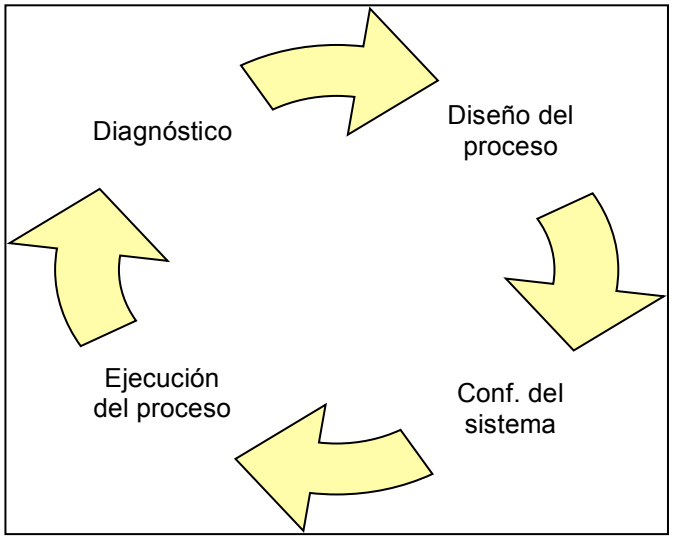


Figura 7: Ciclo de vida BPM

El ciclo de vida de los procesos de negocio en BPM se divide en las siguientes etapas [8]:

- **Diseño del proceso:** En esta etapa, se modelan los procesos actuales como están ("As-Is"), o cómo estos deberían ser ("To-Be"). Estos modelos se realizan con una herramienta BPMS.

- **Configuración de sistema:** En esta etapa se configura la herramienta BPMS y la infraestructura de sistema subyacente (por ejemplo: sincronización de roles y estructura organizacional desde las cuentas de los empleados en un servicio de directorio como LDAP).
  Los procesos de negocio modelados electrónicamente son desplegados en el motor de procesos de la BPMS.

- **Ejecución del proceso:** En esta etapa los procesos son ejecutados en el motor de procesos del BPMS, y son utilizados por los diferentes participantes (sistemas o personas) definidos en el proceso.

- **Diagnóstico:** A través de herramientas de análisis y monitoreo, el analista BPM puede identificar las mejoras necesarias a los procesos de negocio. Esta etapa se enlaza con la etapa de diseño para un nuevo ciclo de mejora.

2.2.7 BPMS

BPMS es la base tecnológica que soporta la implementación de BPM en las empresas.

Según Werke [11], un BPMS es "un sistema de software genérico dirigido por diseños de procesos explícitos para ejecutar y administrar procesos de negocio operacionales".

Según Smith y Fingar [39], "Los BPMS permiten a las empresas modelar, implementar y gestionar los procesos de negocio, que abarcan múltiples aplicaciones empresariales, departamentos, y 'partners', detrás de los cortafuegos y sobre Internet. Los BPMS son una nueva categoría de software y abren una nueva era en la infraestructura de las TI".

Los BPMS pueden ser vistos de dos maneras: como una nueva plataforma sobre la cual se construirá la próxima generación de aplicaciones de negocio, o como una nueva capacidad profundamente embebida en las categorías existentes de aplicaciones de negocio.

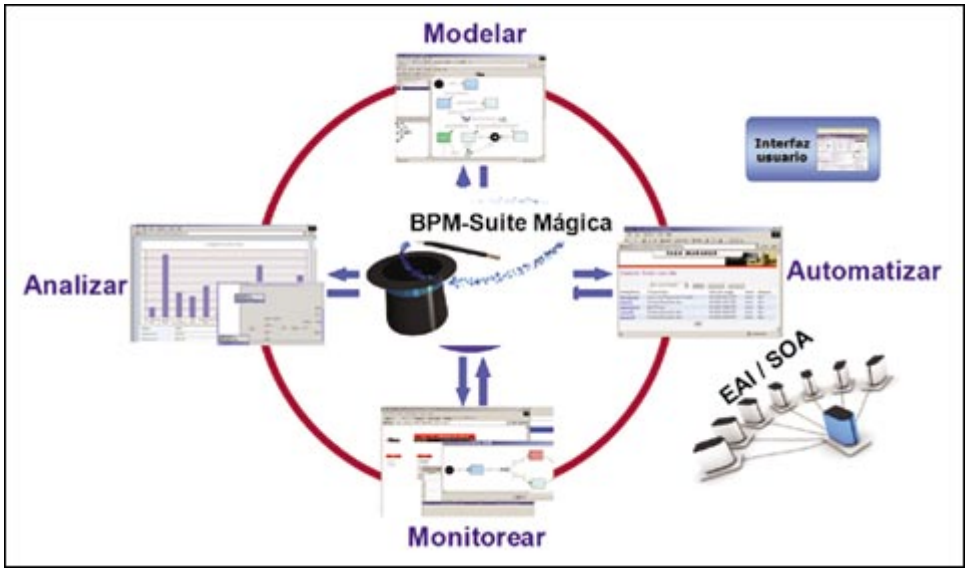


Figura 8: Circulo mágico de las BPM Suite

El estado de la práctica, según Bernhard Hitpass [27], muestra que el ciclo de mejora de procesos vendido por los proveedores de BPMS, el cual el llama el circulo mágico de la BPM suite, actualmente no pasa de ser buenos deseos.
En teoría las BPMS entregan el apoyo suficiente para que la empresa, y en particular para que las funciones vinculadas al negocio, modelen y realicen directamente las modificaciones necesarias a los procesos, agilizando los cambios con una intervención mínima de TI.
En la práctica, Hitpass advierte que es una visión hacia donde se quiere llegar en el futuro, pero que actualmente está lejos de ser una realidad, y que las falsas expectativas que despiertan los proveedores a través del marketing provocan frustración y desilusión sobre la metodología BPM [27].
El problema es creer que los modelos de negocio ejecutados por los motores de procesos puedan ser modelados directamente por los usuarios del negocio. La automatización, requiere especificación en detalle, como modelos de datos, casos de error, condiciones de correlación, instanciación de procesos, etc. Estos detalles no son adecuados para usuarios de proceso y además entorpecen la comunicación a ese nivel, por lo que es necesario distinguir entre modelos de negocio y modelos técnicos.

2.2.8 El estándar BPMN

En 2001 BPMI.org comenzó a desarrollar BPML (Business Process Modeling Language, un lenguaje de ejecución de procesos basado en XML, que luego fue reemplazado por BPEL [3] [34]) y se dio cuenta de la necesidad de una representación gráfica. Se decidió entonces que esta notación estuviera orientada a los usuarios de negocio, los diferentes participantes de BPMI se plantearon el objetivo de consolidar los principios subyacentes en el modelamiento de procesos, acordando una sola notación que otras herramientas y usuarios pudieran adoptar. Esta notación es BPMN, cuya primera versión fue liberada en Mayo de 2004, y que actualmente ya está en su versión 2.0 (liberada oficialmente en Marzo de 2011). Con esta notación es posible crear modelos orientados a negocio, y también modelos técnicos para la ejecución de procesos en un WfMS o un BPMS. [35]

El estándar BPMN es uno de los avances más importantes en la materialización de los modelos de negocio y su ejecución a través de motores de proceso.

BPMN cubre básicamente tres tipos de procesos: procesos privados (ejecutables como no ejecutables), procesos públicos, y coreografías.
Ejemplo de los procesos de negocio que pueden ser modelados con BPMN son:

- Actividades de proceso de alto nivel no ejecutables
- Procesos de negocio ejecutables detallados
- Procesos de negocio “As-Is” o antiguos.
- Procesos de negocio “To-Be” o nuevos.
- Una coreografía: Descripción del comportamiento esperado entre dos o más participantes de negocio.
- Procesos de negocio privados detallados (ejecutables o no), con interacciones con una o más entidades externas(procesos “Black Box”)
- Dos o más procesos ejecutables detallados interactuando.
- Relación detallada de un proceso de negocio ejecutable con una coreografía.
- Dos o más procesos públicos.
- Relación de un proceso público con una coreografía.
- Dos o más procesos de negocios ejecutables interactuando a través de una coreografía.

El estándar ha tenido buena aceptación en la industria y actualmente existen más de 73 implementaciones de BPMN según OMG.

CAPÍTULO 3 ANÁLISIS Y CRÍTICA DE LA SITUACIÓN ACTUAL

Para realizar el análisis se recurrió a tres fuentes de información. El sistema de ticket utilizado para dar seguimiento a los casos en mesa de ayuda. La documentación existente de los procesos del departamento de desarrollo, y de atención a clientes, y la opinión de personas involucradas en la realización de mantenciones correctivas.

Los requerimientos llegan a la mesa de ayuda desde el cliente a través de un llamado telefónico. El operador de mesa de ayuda obtiene información respecto del requerimiento del cliente, lo analiza y entrega asistencia en línea, dando guías generales respecto de la operación correcta del sistema, y gestiona problemas administrativos del sistema, autónomamente o con la ayuda de personal de las áreas de venta, sistemas, proyectos o desarrollo (Figura 9). En este caso se atiende ad-hoc cada instancia y no hay otras definiciones al respecto.

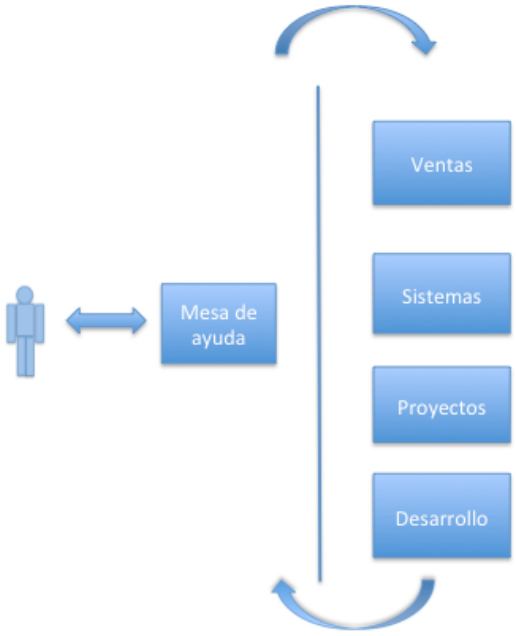


Figura 9: Atención de problemas

En particular, si el operador detecta que el requerimiento del cliente tiene como componente principal un problema en el software que presta el servicio, recaba datos con el cliente, y si lo considera necesario, deriva el requerimiento a los administradores de sistemas para que realicen un chequeo logs y realizar un diagnóstico más acabado. Una vez obtenida toda la información, lo deriva al departamento desarrollo para que se gestione la corrección en caso de ser necesario. Esta actividad en lo formal no se encuentra definida, pero se lleva a cabo utilizando algunas prácticas del proceso de desarrollo como se ve en la Figura 10.

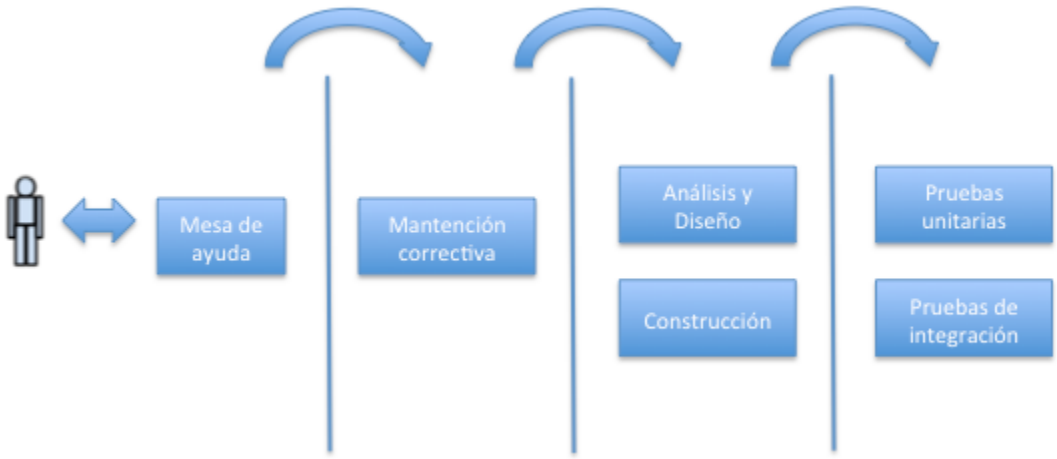


Figura 10: Gestión de corrección de software

Para realizar la gestión de las tareas involucradas en los procesos de gestión de reclamos, y mantención de sistemas, se utilizan las siguientes herramientas.

- **Double Chocolate (DCL)**: Sistema de tickets para el manejo de reclamos de cliente, y actividades de procesos. El uso de esta herramienta es amplio, el flujo de las tareas se realiza de modo manual, siendo responsabilidad de los usuarios el dirigir el trabajo al próximo eslabón en la resolución de problemas y/o requerimientos.

- **Correo electrónico**: Hay tareas cuya coordinación se lleva a través de correos electrónicos. Las más críticas se han migrado a la herramienta DCL para llevar un registro de su ejecución.

- **Procesos documentados**: Existe documentación formal de los procesos principales, en documentos word y diagramas bpmn 1.0, los cuales son la guía para los participantes de los procesos, quienes aplican estas definiciones sobre la herramienta DCL o sobre sistemas de correo.

- **Planillas de cálculo**: Para llevar registro de las actividades realizadas dentro de algunos procesos.

En la práctica estas herramientas han demostrado ser útiles para llevar a cabo las tareas, pero poco eficientes para la ejecución, control y mejora de los procesos. La participación de los usuarios va más allá de ser un eslabón en la producción de un servicio, pues deben interpretar el proceso, resultando en frecuentes errores durante su ejecución.

Al realizar una inspección del sistema de tickets utilizado para hacer seguimiento de casos, se aprecia que existen demoras de hasta 102 horas.

Al consultar a las personas a quienes se les asigna estas tareas, estas responden que tienen compromisos con desarrollos de funcionalidades nuevas en los productos, siendo en la práctica ellos quienes priorizan, bajo su propio criterio, las tareas que realizan primero.

Lo anterior muestra una competencia de recursos, para labores de desarrollo de funcionalidades nuevas y realización de mantenciones, lo cual esta fuertemente influenciado por una **poco clara asignación de los recursos para la atención de los problemas de los clientes actualmente en producción.**
También se comprobó casos en que hubo **clientes esperando meses por una solución, sin ser atendidos los errores con la celeridad requerida**.

Al revisar algunos casos del sistema de tickets se aprecia un flujo no estructurado y adhoc a cada instancia. Tanto al consultar con el encargado de mesa de ayuda como con otros involucrados se aprecia el reconocimiento de algunas tareas generales, y diferencias de opinión respecto de los roles y responsabilidades.

Al revisar la documentación de los procesos documentados en el sistema de gestión de la calidad de la empresa, se aprecia poco detalle al especificar la tarea de análisis en la mesa de ayuda, y una nula mención de el proceso de mantención.

Al hacer estas comprobaciones se apreció una **metodología poco clara para abordar la gestión y asignación de recursos, las que se hacen ad-hoc a cada caso**.

En la documentación existente en el Sistema de gestión de la calidad se observó una **definición poco clara de la cadena de resolución de problemas**.

También se observó confusión en el entendimiento de roles y responsabilidades de cada uno de los participantes del proceso, lo que en algunos casos da como resultado una **mala coordinación al llevar a cabo la atención de los requerimientos de software de clientes.**

La implementación actual del esquema de operación carece de una clara definición de las responsabilidades en las etapas de diagnóstico y análisis de los problemas. También se carece de formalización de la función de mantención, como se detalla en 4.1.1.1.

Desde el punto de vista de las prácticas de desarrollo y mantención del software. Se manifiestan a lo menos tres de los problemas técnicos descritos en 2.1.1 (Problemas técnicos). Estos problemas representan oportunidades de mejora importantes, y son los siguientes:

- En la actualidad las pruebas de los desarrollos se realizan de manera manual, sin ningún tipo de automatización.
- No existen revisiones de código, y este rara vez posee algún tipo de comentario que ayude a su entendimiento futuro. Esto genera una reacción lenta ante las necesidades de cambio durante la mantención, y un retrabajo al tener que recabar la información respecto de las reglas que gobiernan el software en producción. Además representa un retroceso respecto de las prácticas de automatización de pruebas introducidas en algunos software de la empresa durante 2002, y que fueron abandonadas junto con el cambio de software en que se utilizaban, y la rotación de personal.
- Existe duplicidad de código entre diferentes sistemas.
- El desarrollo de WebUI se hace a bajo nivel, sin el apoyo de frameworks. Existe poca experiencia en esta área, siendo cuello de botella en algunos sistemas.

CAPÍTULO 4 SOLUCIONES DESARROLLADAS

A partir de las oportunidades de mejora analizadas en el CAPÍTULO 3 , se desarrollaron dos soluciones tendientes a mejorar la eficiencia en la solución de problemas y la mantención de software.

1) Automatización del flujo del proceso de mantención y gestión de problemas, y agilización del cambio de este proceso.

    - Automatización de flujos de gestión de problemas y mantención de software: El objetivo es hacer más fluido el flujo de trabajo y ser consistente con la definición de las tareas del proceso, disminuyendo los errores, y posibilitando el registro y gestión de indicadores de los procesos.
    - Agilización del cambio de procesos: El objetivo es proveer de herramientas que permitan agilizar el cambio de los procesos.

2) Definición de la organización de mantenimiento de software, y mejoramiento de procesos y prácticas involucradas en la mantención del software.

    - Cambios y formalización de procesos de gestión de problemas, desarrollo, y mantención de software: El objetivo es mejorar la comunicación y entendimiento de las responsabilidades entre las partes involucradas, como también agilizar los procesos.
    - Definición de una organización para la mantención del software: El objetivo es mejorar la disponibilidad de recursos para las tareas de mantención de software y disminuir el impacto de nuevos proyectos de desarrollo en la atención de los clientes de los servicios en producción.
    - Mejoramiento de prácticas relacionadas con la mantención de software: El objetivo es facilitar el traspaso de conocimiento, y la detección temprana de errores, mejorando la calidad intrínseca de los productos de software.

Principales Ventajas Esperadas

- Mejorar la comunicación inter-áreas en el ámbito de la atención de problemas de clientes.
- Obtener una base de procesos clara sobre la cual seguir mejorando los procesos de atención a clientes.
- Asegurar la disponibilidad de recursos para las tareas de mantención de software.
- Establecer un proceso de mejora continua de los procesos de atención y gestión de problemas.
- Mejoras a las prácticas en ámbito del desarrollo y mantención del software.
- Soporte para la automatización de flujos

- Establecimiento de estándar para la definición y automatización de procesos (BPMN).
- Soporte para el cambio evolutivo de los procesos de manera ágil a partir de estándar BPMN
- Obtención de herramienta con acceso a métricas para la medida y mejora de los procesos.
- Automatización de flujos de proceso, de modo que el proceso guie al usuario en su realización.
- Software base para automatización de código abierto, soportado por la comunidad de internet, con licencia apache.
- Software base para automatización con soporte para autenticación Ldap lo que permite centralizar y facilitar la gestión de usuarios mediante los sistemas actuales de la empresa.

Principales Desventajas

- La parte relativa a la automatización, al ser una implementación propia basada en librerías de código abierto, requiere de mantención en la medida que requiera agregar nuevas características al software.
- La herramienta de automatización propuesta es tan flexible como la definición del proceso definido, por lo que las atenciones adhoc no previstas no serán posibles en el sistema.
- La herramienta de automatización depende en gran medida del proyecto de código abierto del motor bpm utilizado [4.1.3.2.1], por lo que depende de la estabilidad y mantenimiento de la comunidad de dicho proyecto.

Para hacer frente a la mantención de la plataforma de automatización, se debe definir el manual de mantención, según se propone para todos los software en [4.2]. Limitar el ámbito de este software solo en aspectos generales de proceso, sin involucrarlo en procesos particulares, los cuales deben cubrirse en la definición BPMN.

Para hacer frente a la poca flexibilidad de las definiciones de proceso se puede definir procesos adicionales, o el mismo proceso actual, para que se encarguen de las excepciones. De modo de atender la mayoría de los casos con los procesos definidos formalmente, y que los casos restantes sean de análisis posterior, para mejorar los procesos formalmente definidos.

Para hacer frente a una eventual perdida de mantención del proyecto open source base, se debe migrar a otro proyecto compatible [4.1.3.2.1]. Al ser motores basados en el estándar BPMN, la migración no debe presentar dificultad técnica, más allá de las horas hombre dedicadas a la migración.

Alternativas de solución

Es claro que se deben formalizar y aclarar los procesos de la empresa, mejorar las prácticas que afectan a la mantención del software, como también, establecer y asegurar los recursos para la mantención del software.

Las diferencias en la solución pueden estar en las herramientas de apoyo de TI utilizadas.

- Se puede llevar a cabo utilizando la herramienta de gestión de tickets utilizada actualmente para la gestión de problemas, usando flujos adhoc, sin agilizar el cambio de procesos a través de BPMN como se establece en esta propuesta. Esto ciertamente mejora la atención de clientes, pero eleva el riesgo de errores de los participantes en el proceso, cuya medida de mitigación puede estar dada por actividades de capacitación.

- Se puede utilizar una herramienta basada en un proyecto opensource de motor de flujos que cumpla con el estándar BPMN, lo que permite apoyar la automatización del flujo del proceso, y el cambio de los procesos de manera ágil, obteniendo los beneficios mencionados anteriormente en “Principales Ventajas”.

En este trabajo se ha escogido la opción de apoyar con una herramienta basada en un motor bpm, debido a los beneficios que ofrece la utilización de tecnologías de bpm en una organización con alto grado de cambio como es el caso.

Por otro lado, respecto a las prácticas de desarrollo y mantención de software, se debe destacar que la implementación de pruebas automatizadas, inspirada en TDD, no es desconocida para la empresa, y fueron de gran ayuda en el desarrollo de sus primeros sistemas. Permitieron tener una documentación viva del sistema en desarrollo/mantención, dejando en el código de pruebas, gran cantidad de conocimiento del negocio, lo que fue muy útil para los desarrolladores que tuvieron que intervenir el código tiempo después.

Se hace importante, de este modo, reincorporar las prácticas de TDD como parte del proceso de mantención y hacerlas extensivas al proceso de desarrollo, cómo herramienta de apoyo para el mejoramiento de la calidad del código producido, agilización de pruebas de software, y mantenimiento de este.

4.1 MEJORAS AL PROCESO DE MANTENCIÓN Y GESTIÓN DE PROBLEMAS, Y AUTOMATIZACIÓN DE SU FLUJO

4.1.1 Gestión de problemas y mantención correctiva

4.1.1.1 Proceso actual

El proceso actual para la gestión de problemas es conocido internamente como “Soporte de usuarios” y abordado en el proceso “Atención de requerimientos de clientes”.

El soporte de usuarios es llevado a cabo por la mesa de ayuda, y sus actividades típicas son la asistencia en el uso de los sistemas y la gestión de problemas en la prestación de los servicios contratados (Figura 11).

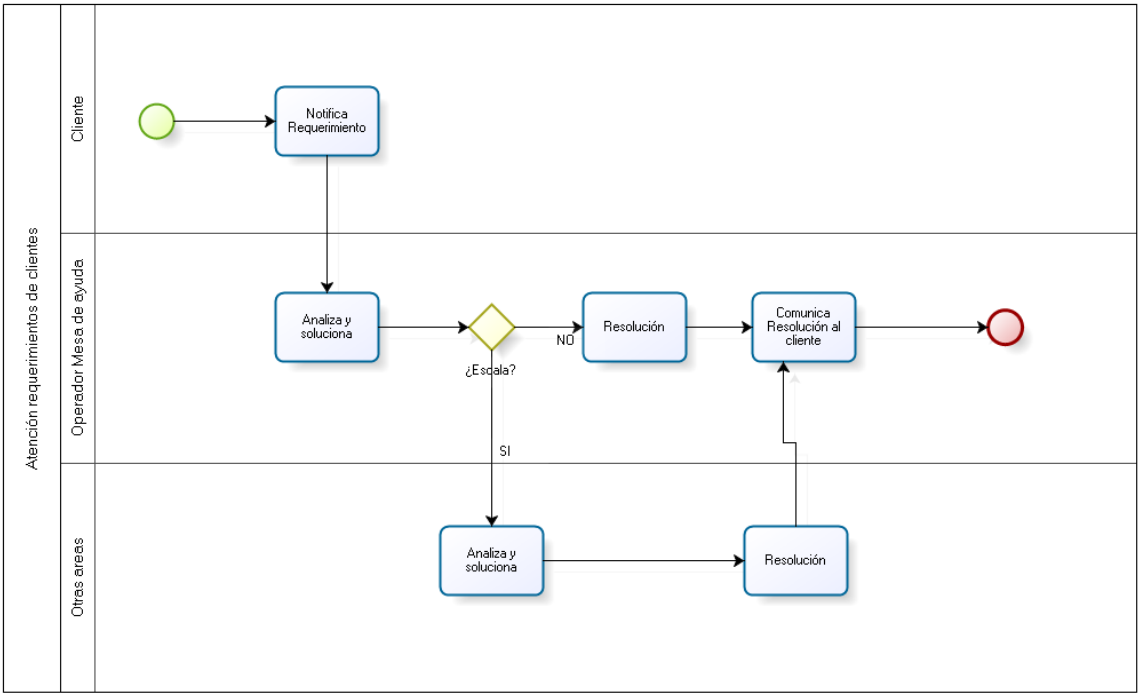


Figura 11: Proceso de atención de requerimientos de clientes

Este proceso se concentra en el registro del problema del cliente, y su solución en línea. No hay mención explicita sobre los problemas de software que pueden reportar los clientes y cómo estos se gestionan. Sólo se hace mención a un escalamiento a otras áreas, sin detallar responsables ni procedimientos claros al respecto.

Actualmente no se encuentra definido formalmente un proceso de mantención correctiva, y las correcciones, de llevarse a cabo, se realizan de manera ad-hoc en cada caso.

Este proceso actualmente se encuentra soportado por un sistema de tickets como se mencionó en el CAPÍTULO 3 . De este sistema se exportaron los registros correspondientes a tres meses de operación y se generaron indicadores de rendimiento para el proceso actual.
En la siguiente tabla se aprecian las estadísticas actuales del proceso.

	Meses		
	2012-12	2012-11	2012-10
Tiempo promedio(horas)	1.42	1.39	1.45
Tiempo máximo(horas)	198.48	191.083	203.25

Tabla 2: Datos de rendimiento proceso actual

4.1.1.2 Mejoras propuestas

El reporte de problemas actualmente se lleva a través del sistema de ticket, correo electrónico, y otros medios informales. Sin embargo no hay una estructura normada para estos reportes, lo que dificulta su gestión.

Según la experiencia práctica y lo descrito en la bibliografía, una vez que el software está en producción, las actividades típicas durante la operación del software son:

- Soporte de usuarios
- Reporte de problemas

El proceso propuesto para la gestión de problemas y mantención correctiva considera dos niveles de mesa de ayuda, y las siguientes actividades:

1. El primer nivel de mesa de ayuda asiste al cliente en el uso y solución de problemas con los sistemas y servicios que otorga la empresa.
    a. Obtiene antecedentes del problema
    b. Realiza análisis del problema
    c. Genera una resolución
2. El segundo nivel de mesa de ayuda tiene un conocimiento más avanzado en la arquitectura de los sistemas, asistiendo al cliente en los problemas más especializados.
    a. Investiga problema

b. Realiza análisis del problema
c. Genera resolución

3. El segundo nivel además evaluará la necesidad de generar reportes de problemas de software en caso de que el origen sea un error en el software.
4. El DRM es aprobado/rechazado por el Jefe de Mantenimiento.
5. El Ingeniero de mantenimiento realiza
   a. Análisis
   b. Diseño e Implementación
   c. Actualización de ítems de configuración.
6. Se realiza el paso a producción del producto.

Estás actividades son la alternativa de mejora al proceso original mostrado en la Figura 11, y se organizan en dos procesos. Estos procesos son dependientes de dos áreas distintas, y su relación se muestra en la Figura 12 y Figura 13.

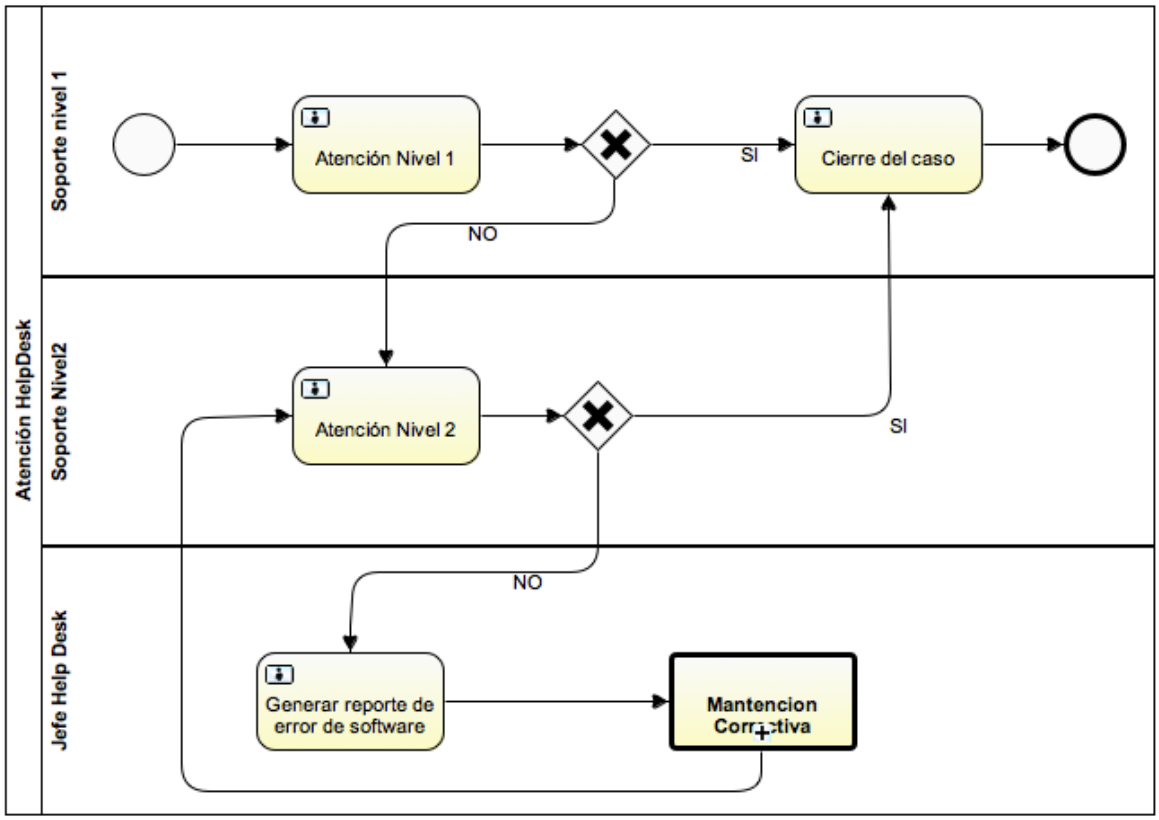


Figura 12: Flujo de proceso propuesto para atención HelpDesk(integrado con mantención correctiva)

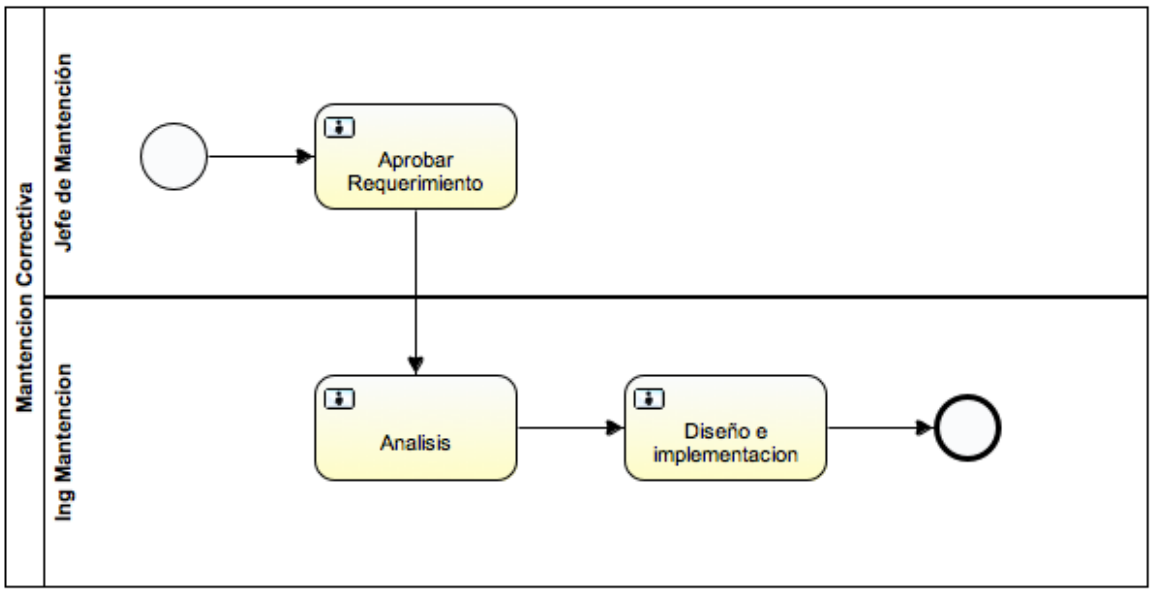


Figura 13: Flujo de proceso propuesto para mantención correctiva

Es clave la incorporación de un segundo nivel de mesa de ayuda, especializado en los productos que brinde apoyo en la resolución de problemas al nivel uno, y que filtre los problemas que son de fallas en el software de aquellos que corresponden a una incorrecta utilización por parte de los usuarios.

Otra de las definiciones importantes en el proceso de gestión de problemas y de mantención correctiva es la definición formal del reporte de falla y qué datos debe contener, los que se listan a continuación:

- Producto: Nombre que identifica el producto que presenta el problema
- Versión: Número de versión del producto que presenta el problema
- Plataforma: Tipo de máquina en el que se presenta el problema (Pc, Mac, Hp, Sun, Smartphone, TabletPC, Otro)
- Sistema operativo: Nombre que identifica al sistema operativo sobre el cual se presenta el problema.
- Descripción: Pasos para reproducir el error, resultado esperado, y resultado obtenido.
- Severidad: Severidad del problema
    - Crítica: El sistema se “cae”, pierde datos, agota la memoria.
    - Mayor: Pérdida de funcionalidad mayor
    - Normal: Alguna pérdida de funcionalidad bajo situaciones específicas
    - Menor: Pérdida de funcionalidad menor, u otro problema con un camino alternativo simple
    - Trivial: Problema cosmético, como falta de ortografía o alineación de elementos.
- Prioridad: Prioridad para resolver (Baja, Normal, Alta, Urgente, Inmediato)

4.1.1.3 Simulación

La simulación de procesos de BPM es utilizada para la validación de las modificaciones al proceso. Esto se corresponde a la etapa de diseño de proceso descrita en 2.2.6. En este contexto, la simulación es una alternativa a implementar directamente el ciclo de mejora propuesto en 2.2.6, logrando, a través de simulaciones sucesivas, un modelo que cumpla con los resultados de rendimiento esperados.

En particular, se valida el proceso de gestión de problemas y mantención correctiva, a través de una herramienta de simulación basada en modelos de procesos en BPMN. Estás herramientas generan como salida un informe con tiempo máximo, mínimo, y promedio de los procesos, además de una medida de la utilización de los recursos.

Además de validar el proceso, los datos obtenidos de la simulación son utilizados como base para determinar los costos de implementación descritos en 4.3.

La selección de la herramienta utilizada en la simulación se describe en el anexo G. A continuación, se muestra el escenario de simulación considerado y el resultado de las simulaciones realizadas con la herramienta de simulación.

4.1.1.3.1 Escenario de simulación.

Para comprobar el impacto de los cambios propuestos se realiza la simulación del proceso anterior versus el nuevo. Para esta simulación se establece el siguiente escenario, basado en datos obtenidos del sistema de tickets (ver CAPÍTULO 3), y juicio experto de los involucrados en las tareas de mantención:

- Los requerimientos llegan según una distribución exponencial con media 4.88 minutos.
  Este supuesto se basa en una muestra tomada del sistema de tickets descrito en CAPÍTULO 3. Al hacer una gráfica del tiempo entre la creación de tickets, se obtiene la siguiente distribución empírica:

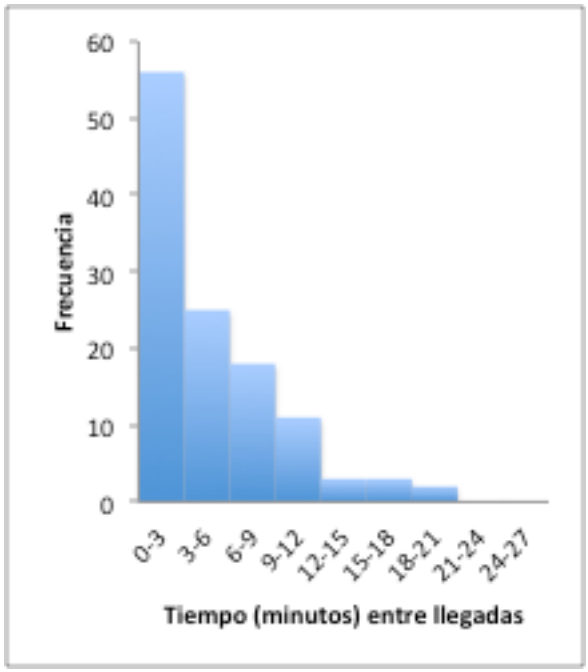


Figura 14: Distribución de requerimientos para una muestra de un día

Esta distribución se aproxima bastante a la forma de la distribución exponencial. Lo anterior es consistente con el uso de la distribución exponencial para simular centros de llamado, y la llegada de clientes en teoría de colas según expresado en la literatura [42]

- Llegan en promedio 110 requerimientos al sistema por día.

Caso 1: Proceso actual

Como se vio en 4.1.1.1 este proceso posee dos caminos alternativos de ejecución.

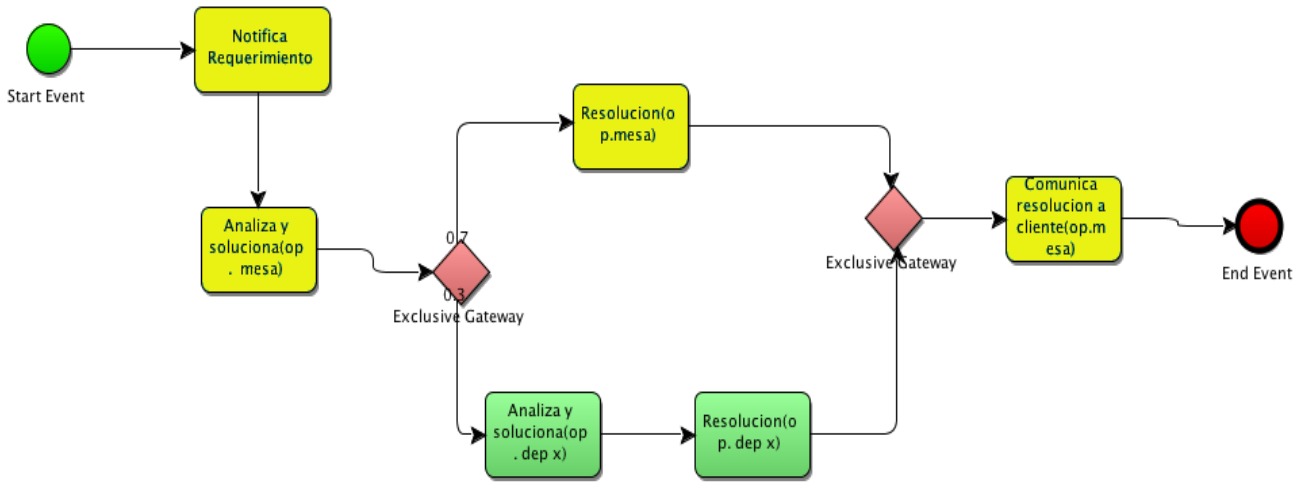


Figura 15: Path1, Solución de problema en mesa de ayuda

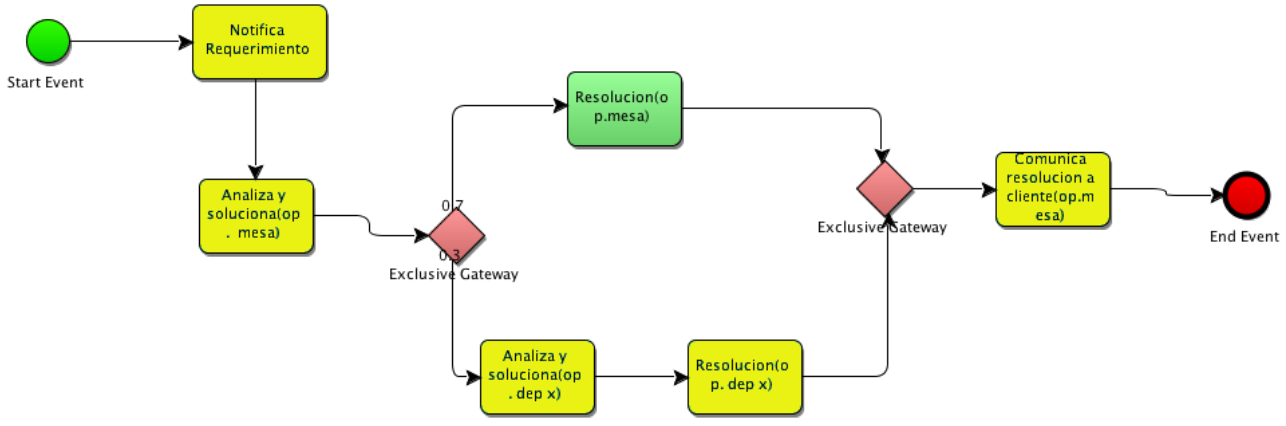


Figura 16: Path2, Solución de problema escalando a otro departamento

Tarea	Rol encargado	Tiempo
Notifica Requerimiento	Operador mesa	5 minutos
Analiza y soluciona	Operador mesa	5 minutos, max:10 minutos
Resolución	Operador mesa	10 minutos
Analiza y soluciona	Apoyo otro depto.	4 horas, 50 minutos,
Resolución	Apoyo otro depto.	10 minutos, max:11 minutos
Comunica resolución	Apoyo otro depto.	10 minutos

Tabla 3: Parámetros tareas de proceso actual

- El 85% de los casos se resuelven en mesa de ayuda
- El 15% de los casos se derivan a otro departamento, por lo general a desarrollo.
- En el proceso actual participan 8 operadores de mesa de ayuda
- En el proceso actual participan 2 personal de apoyo de otro departamento.

Caso 2: Proceso propuesto

Como se vio en 4.1.1.2 este proceso posee tres caminos alternativos de ejecución.

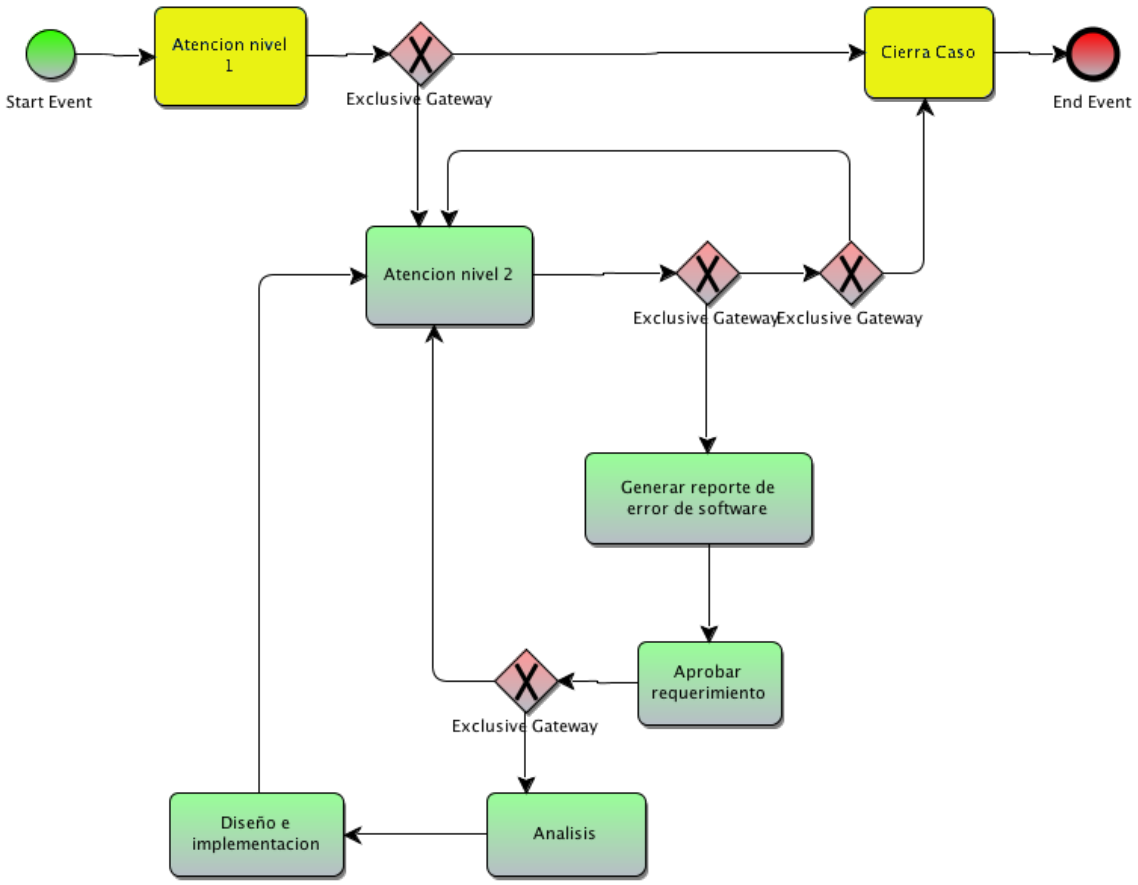


Figura 17: Path1, Solución de problema en nivel 1 de mesa de ayuda

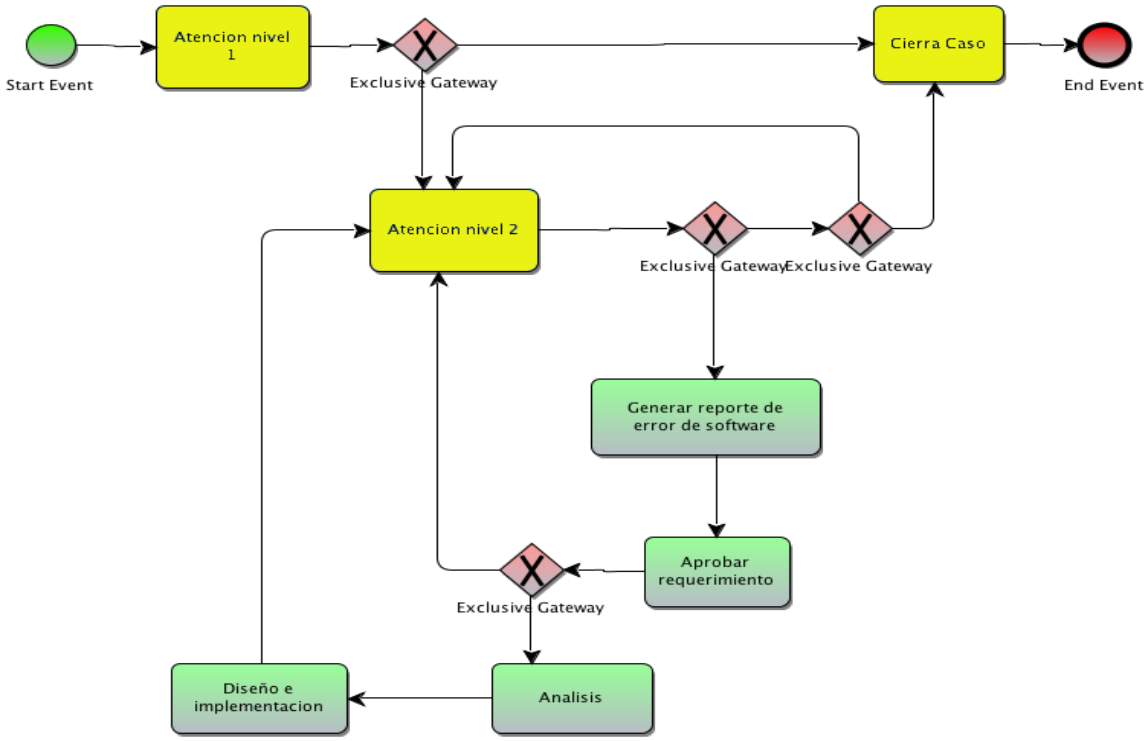


Figura 18: Path2, Solución de problema en nivel 2 de mesa de ayuda

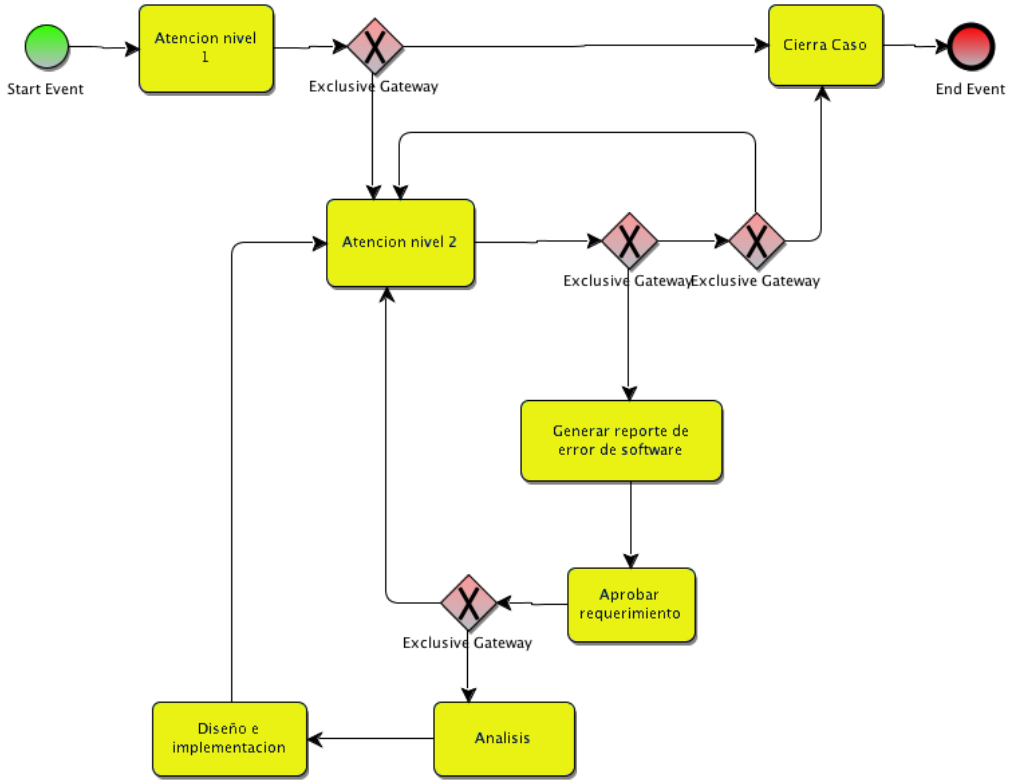


Figura 19: Path3, Solución de problema ejecutando mantención correctiva

Tarea	Rol encargado	Tiempo
Atención nivel 1	Operador nivel 1	15 minutos
Cierra caso	Operador nivel 1	10 minutos
Atención nivel 2	Operador nivel 2	51 minutos, max:66 minutos
Generar reporte de error de software	Operador nivel 2	3 minutos, max: 6 minutos.
Aprobar requerimiento	Jefe de mantención.	5 minutos, max:10 minutos
Análisis	Ingeniero de mantención	50 minutos, max: 1 hora, 40 minutos.
Diseño e implementación	Ingeniero de mantención	4 horas, max: 6 horas.

Tabla 4: Parámetros tareas de proceso propuesto

- El 85% de los casos se resuelve en Atención de nivel 1
- El 15% de los casos se derivan a nivel 2.
- De los casos de nivel 2, 90% son solucionados en este nivel, y 10% se deben a errores de software, iniciándose una instancia de mantención correctiva.

4.1.1.3.2 Validación de la simulación del proceso actual

Para validar los datos obtenidos de la simulación del proceso actual se ejecutó la simulación según los parámetros especificados en 4.1.1.3.1 y se comparó con información histórica del proceso.
La obtención de información histórica estuvo restringida por la disponibilidad de detalle de las tareas en el sistema para este fin, pudiendo obtener solo información confiable respecto a la duración total de cada instancia de proceso. Dado lo anterior la validación se hizo contra los datos de rendimiento de proceso actual mostrados en la Tabla 2: Datos de rendimiento proceso actual.

	Resultado simulación	Datos rendimiento proceso actual por mes			Desviación porcentual promedio
		2012-12	2012-11	2012-10	
Tiempo promedio(horas)	1.38	1.42	1.39	1.45	2.9%
Tiempo máximo(horas)	191.53	198.48	191.083	203.25	3.6%

Tabla 5: Simulación versus datos reales

La simulación arrojó un tiempo promedio cercano a 1.4 horas , y un tiempo máximo cercano a las 200 horas. Al observar los datos de rendimiento del proceso actual, estos se asemejan bastante a resultados de la simulación, por lo que se puede decir que la simulación se acerca de manera aceptable a la realidad.

4.1.1.3.3 Resultado de simulación.

Se realiza simulación de 4 semanas de operación del sistema, obteniendo los siguientes resultados.

Caso 1: Proceso actual.

A continuación se muestra un resumen de las simulaciones realizadas para este caso.

Tiempo de simulación	4 semanas de simulación
	100 requerimientos por día
Distribución	Exponencial negativa, media 4.88 minutos

T.Pr. Proc	Tiempo promedio del proceso
T.Max Proc	Tiempo máximo de una instancia del proceso
T.Pr. Path1	Tiempo promedio de casos resueltos en mesa de ayuda
T.Pr. Path2	Tiempo promedio de casos resueltos con escalamiento

					Recursos			Utilización de recursos	
Nº simulación	**T. Pr. Proc**	**T. Max Proc**	**T. Pr. Path1**	**T.Pr. Path2**	**Op. Depto X**	**Op. Mesa**	**OBS**	**Op. Depto X**	**Op. Mesa**
1	1,38	191,53	0,56	179,02	2	8	Op. Dept X media jornada	99,48%	69,62%
2	3,12	168,67	0,56	122,51	2	8	Op. Dept X jornada completa	99,33%	69,97%
3	4,59	50,87	0,56	34,01	4	8	Op. Dept X jornada completa	99,07%	71,73%
4	1,44	14,31	0,57	6,69	5	8	Op. Dept X jornada completa	88,61%	72,18%

Tabla 6: Resumen simulaciones proceso actual

De los resultados de las simulaciones se aprecia que el mayor problema se encuentra en el escalamiento de los casos (T.Pr Path2), para el cual se obtuvo tiempos de servicios de 179,02Hrs promedio. El recurso “Op. Depto X” se aprecia con utilización sobre el 99%, lo que tiene como consecuencia que los requerimientos se comenzarán a encolar hasta que el recurso esté disponible.

En las siguientes simulaciones se aumenta el número del recurso “Op. Depto X”, hasta obtener un porcentaje de utilización de 88,61%. De esta manera se logran los objetivos para el proceso, utilizando una configuración de 8 operadores de mesa, y 5 operadores de otro departamento (desarrollo). Los costos asociados a esta solución, son de $9,117,126.58, lo que es demasiado elevado dado los costos asociados a los recursos según se aprecia en la tabla de costos siguiente.

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador mesa	8	69,62%	$ 2.539.812,46
1	operador x	2	99,48%	$ 1.456.410,38

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador mesa	8	69,97%	$ 2.552.354,36
2	operador x	2	99,33%	$ 2.908.477,56

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador mesa	8	71,73%	$ 2.616.861,26
3	operador x	4	99,07%	$ 5.801.439,16

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador mesa	8	72,18%	$ 2.633.069,78
4	operador x	5	88,61%	$ 6.486.056,80

Tabla 7: Costos asociados a cada configuración de recursos

Caso 2: Proceso propuesto.

A continuación se muestra un resumen de las simulaciones realizadas en este caso.

Para llegadas de requerimientos según distribución exponencial con media 4.88 minutos.

Tiempo de simulación	4 semanas de simulación
	100 requerimientos por día
Distribución	Exponencial negativa, media 4.88 minutos

T.Pr. Proc	Tiempo promedio del proceso
T.Max Proc	Tiempo máximo de una instancia del proceso
T.Pr. Path1	Tiempo promedio de casos resueltos en nivel 1
T.Pr. Path2	Tiempo promedio de casos resueltos en nivel 2
T.Pr. Path3	Tiempo promedio de casos resueltos en nivel 3 (mant. correctiva)

						Recursos				Utilización de recursos			
Nº simulación	T. Pr. Proc	T. Max Proc	T. Path1	T. Path2	T. Path3	Op. N1	Op. N2	Ing. Mant.	Jefe Mant.	Op. N1	Op. N2	Ing. Mant.	Jefe Mant.
1	4,12	105,29	0,45	42,33	94,88	7	1	1	1	68,45%	99,75%	19,39%	0,39%
2	1,10	49,31	0,56	2,69	25,59	6	2	1	1	82,02%	86,02%	72,21%	1,52%

Tabla 8: Resumen simulaciones proceso propuesto

Se inicia la simulación con 1 operador de nivel 2 y 1 ingeniero de mantención, obteniéndose tiempos promedio mejores para la mantención correctiva, pero un porcentaje de utilización de "Op. N2" que sugiere encolamiento de los

requerimientos en la tarea asociada. En la segunda simulación se establece que la configuración mínima para lograr obtener una performance según los objetivos planteados, con 2 operador de nivel 2 y 1 ing. de mantención, ambos a tiempo completo. 7 operadores de nivel 1, y 1 jefe de mantención, con un costo total de $4,279,502.68.

Los costos asociados a esta configuración son:

N º Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	7	68,45%	$2.185.025,46
	operador nivel 2	1	99,75%	$550.605,28
	ingeniero mantencion	1	19,39%	$283.808,60
1	jefe mantencion	1	0,39%	$7.416,52

N º Simulación	Resource	Cantidad	Utilización	Costo
	operador nivel 1	6	82,02%	$2.244.031,86
	operador nivel 2	2	86,02%	$949.650,22
	ingeniero mantencion	1	72,21%	$1.057.094,62
2	jefe mantencion	1	1,52%	$28.725,98

Tabla 9: Costos asociados a cada configuración de recursos

Considerando que se espera un aumento significativo en el número de clientes, y que los casos aumentarán en directa proporción a este aumento, se realiza una simulación considerando un aumento del 100% en los requerimientos para mesa de ayuda. Es decir 200 requerimientos por día.

Tiempo de simulación	4 semanas de simulación
	100 requerimientos por día
Distribución	Exponencial negativa, media 4.88 minutos

T.Pr. Proc	Tiempo promedio del proceso
T.Max Proc	Tiempo máximo de una instancia del proceso
T.Pr. Path1	Tiempo promedio de casos resueltos en nivel 1
T.Pr. Path2	Tiempo promedio de casos resueltos en nivel 2
T.Pr. Path3	Tiempo promedio de casos resueltos en nivel 3 (mant. correctiva)

						Recursos				Utilización de recursos			
Nº simulación	T. Pr. Proc	T. Max Proc	T. Path1	T. Path2	T. Path3	Op. N1	Op. N2	Ing. Mant.	Jefe Mant.	Op. N1	Op. N2	Ing. Mant.	Jefe Mant.
1	52,81	140,20	61,29	79,31	117,82	6	2	1	1	99,95%	99,76%	69,64%	1,47%
2	40,19	120,53	45,44	47,18	88,02	7	3	1	1	99,93%	90,97%	89,93%	3,04%
3	27,71	115,72	29,62	34,23	80,15	8	3	1	1	99,92%	97,85%	91,08%	3,04%
4	15,01	95,56	14,30	24,55	73,22	9	3	1	1	99,91%	99,27%	91,33%	2,88%
5	4,48	90,70	2,49	17,29	66,59	10	3	1	1	98,50%	99,72%	91,35%	2,70%
6	2,72	91,75	0,60	16,04	65,30	11	3	1	1	91,34%	99,74%	91,36%	2,64%
7	2,62	91,71	0,48	15,98	65,21	12	3	1	1	83,77%	99,74%	91,36%	2,64%
8	0,99	90,94	0,48	2,01	62,38	12	4	1	1	84,49%	89,12%	92,70%	4,03%
9	0,88	28,92	0,48	2,01	18,15	12	4	2	1	84,66%	89,12%	84,65%	4,03%
10	0,82	27,74	0,48	1,57	18,51	12	5	2	1	84,68%	71,63%	84,65%	4,03%

Tabla 10 : Resumen simulaciones proceso propuesto escenario futuro

Se inicia la simulación con 1 operador de nivel 1 y 1 ingeniero de mantención, obteniéndose tiempos excesivos para los objetivos planteados.

Durante la simulación se establece que la configuración mínima para logra obtener un rendimiento, según los objetivos planteados, es con 12 operadores de nivel1, 4 operadores de nivel 2 y 1 ing. de mantención a tiempo completo, y 1 jefe de mantención. Sin embargo, al presentar niveles de uso de los recursos cercanos o levemente superiores a 90% y considerando que un 10% del tiempo se dedica a otras actividades, se debe considerar una configuración de 12 operadores de nivel 1, 5 operadores de nivel 2, 2 ing. de mantención y 1 jefe de mantenimiento.

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	6	99,95%	$2.734.573,48
	operador nivel 2	2	99,76%	$1.101.386,74
	ingeniero mantención	1	69,64%	$1.019.545,46
1	jefe mantención	1	1,47%	$27.904,38

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	7	99,93%	$3.189.895,94
	operador nivel 2	3	90,97%	$1.506.469,64
	ingeniero mantención	1	89,93%	$1.316.594,72
2	jefe mantención	1	3,04%	$57.646,30

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	8	99,92%	$3.645.106,30
	operador nivel 2	3	97,85%	$1.620.313,66
	ingeniero mantención	1	91,08%	$1.333.403,88
3	jefe mantención	1	3,04%	$57.646,30

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	9	99,91%	$4.100.265,36
	operador nivel 2	3	99,27%	$1.643.964,56
	ingeniero mantención	1	91,33%	$1.337.134,64
4	jefe mantención	1	2,88%	$54.601,64

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	10	98,50%	$4.491.406,96
	operador nivel 2	3	99,72%	$1.651.375,62
	ingeniero mantención	1	91,35%	$1.337.350,58
5	jefe mantención	1	2,70%	$51.127,22

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	11	91,34%	$4.581.495,08
	operador nivel 2	3	99,74%	$1.651.692,56
	ingeniero mantención	1	91,36%	$1.337.445,74
6	jefe mantención	1	2,64%	$50.079,68

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	12	83,77%	$4.583.945,70
	operador nivel 2	3	99,74%	$1.651.692,56
	ingeniero mantención	1	91,36%	$1.337.489,66
7	jefe mantención	1	2,64%	$50.079,68

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	12	84,49%	$4.623.063,66
	operador nivel 2	4	89,12%	$1.967.659,20
	ingeniero mantención	1	92,70%	$1.357.196,32
8	jefe mantención	1	4,03%	$76.407,22

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	12	84,66%	$4.632.563,66
	operador nivel 2	4	89,12%	$1.967.659,20
	ingeniero mantención	2	84,65%	$2.478.682,54
9	jefe mantención	1	4,03%	$76.407,22

Nº Simulación	Recurso	Cantidad	Utilización	Costo
	operador nivel 1	12	84,68%	$4.633.690,36
	operador nivel 2	5	71,63%	$1.977.024,80
	ingeniero mantención	2	84,65%	$2.478.682,54
10	jefe mantención	1	4,03%	$76.407,22

Tabla 11: Costos asociados a cada configuración de recursos

4.1.1.3.4 Robustez de la simulación

Las variables relevantes para las simulaciones realizadas son el tiempo entre llegadas de requerimientos, y los tiempos asignados a cada tarea.

Para los tiempos entre llegadas, se calculó la distribución empírica del tiempo entre llegadas, resultando una distribución exponencial negativa, que es el estándar para este tipo de problemas. Una alternativa a esto, es considerar una tasa constante de llegadas, lo que da como resultado un aumento en los costos, entre un 19% y 20%, y una configuración distinta a la estimada para la solución.

Tiempo de simulación	4 semanas de simulación
	100 requerimientos por día
Distribución	Constante 4.88 minutos

T.Pr. Proc	Tiempo promedio del proceso
T.Max Proc	Tiempo máximo de una instancia del proceso
T.Pr. Path1	Tiempo promedio de casos resueltos en nivel 1
T.Pr. Path2	Tiempo promedio de casos resueltos en nivel 2
T.Pr. Path3	Tiempo promedio de casos resueltos en nivel 3 (mant. correctiva)

						Recursos				Utilización de recursos			
Nº simulación	**T. Pr. Proc**	**T. Max Proc**	**T. Path1**	**T. Path2**	**T. Path3**	**Op. N1**	**Op. N2**	**Ing. Mant.**	**Jefe Mant.**	**Op. N1**	**Op. N2**	**Ing. Mant.**	**Jefe Mant.**
1	4,84	145,29	0,42	49,16	112,59	7	1	1	1	71,02%	99,67%	45,04%	0,87%
2	1,32	75,77	0,42	2,95	47,75	6	2	1	1	85,12%	89,70%	95,92%	2,35%
3	0,94	20,48	0,42	2,95	12,49	6	2	2	1	85,23%	89,70%	59,21%	2,35%
4	0,69	13,49	0,42	1,51	8,78	6	3	2	1	85,23%	59,80%	59,82%	2,35%

Tabla 12: Simulación con distribución entre llegadas constante

Los tiempos asignados a las tareas de operadores de nivel1 y nivel2 están basados en su mayoría a información histórica extraída del sistema de tickets, mientras que los tiempos asociados a tareas de mantención se determinan utilizando juicio experto de las personas que han participado de estas actividades.

En general se concluye que la simulación realizada es especialmente sensible a la duración estimada para las tareas asociadas a la mantención de software, por lo que se debe tener en cuenta a la hora de implementar las mejoras y adecuar estas estimaciones cuando se cuente con información histórica. También se debe tener en cuenta que la distribución de llegadas es representativa de la situación actual, pero no necesariamente es permanente en el tiempo, por lo que se debe estar atento a cambios en la demanda de atención de los clientes, que pudieran hacer variar esta estimación.

4.1.2 Gestión de cambios (mantención evolutiva, preventiva y perfectiva)

4.1.2.1 Proceso actual

Considerando los tipos de mantención evolutiva, preventiva y perfectiva, presentados en 2.1, los cambios más frecuentes en la organización son los de tipo evolutivo y perfectivo. De los sistemas de la empresa, los más propensos a este tipo de cambios son los que dan soporte a los servicios de factura electrónica. Por otro lado, los sistemas relacionados con el servicio de autoridad certificadora son más estables en su especificación funcional, y de rendimiento.

Este tipo de cambios son generados a partir de cambios en la concepción de los productos desde el punto de vista comercial, cambios en el sistema de factura electrónica por parte del SII, o por cambios en las exigencias de rendimiento de los sistemas.

Actualmente estos cambios se gestionan a través de una solicitud, vía correo electrónico, al gerente de desarrollo, y una vez aprobado, es asignado un equipo o desarrollador a cargo siguiendo las actividades definidas para el ciclo de desarrollo de software de la empresa.

No existe formalmente definido un proceso para gestionar el cambio del software en producción como tampoco una guía de adaptación del proceso de software para el caso de la mantención del software.

4.1.2.2 Mejora propuesta

El proceso propuesto para abordar la gestión de cambios del software considera las siguientes actividades.

1. Solicitud de cambio: Solicitud que incluye el producto, los cambios a introducir, y responsable de la solicitud. Se utiliza la misma estructura del DRU actualmente en uso.
2. Análisis de impacto: Se hace un análisis de los sistemas afectados por el cambio requerido, los recursos involucrados en la actividad, y un análisis de los riesgos que la implementación de los cambios conlleva.
3. Autorización de cambio: Comité de cambio de producto analiza informe de impacto y autoriza los cambios.
4. Actualización de arquitectura: Se actualiza arquitectura del sistema si es necesario según los cambios solicitados en el DRU. El diseño de la arquitectura actualizada se incluye en el DRS.
5. Plazos comprometidos para la mantención. En el DRS se incluyen además los plazos comprometidos en la codificación de lo especificado en el documento.
6. Actualización del plan de pruebas de aceptación. Este documento se genera una vez conocido el DRS, y toma como entrada el último plan de

pruebas del producto, realizando las modificaciones que sean necesarias, de tal forma de incluir comprobaciones para cada uno de los requerimientos de usuario existentes, y los requerimientos de cambio actuales.

7. Actualización de casos y procedimientos de prueba, se actualiza el último plan de pruebas del producto de tal forma de asegurar que las pruebas sean lo suficientemente completas que abarquen y permitan demostrar que se satisfacen cada uno de los requerimientos de usuario. Usualmente esta fase se trabaja en conjunto con el cliente para llegar a acuerdo de lo que se va a probar y obtener de él, el conjunto de datos y procedimientos de prueba que se utilizará.
8. Codificación, Es el proceso de implementación de los cambios del software solicitado en el DRU y especificado en el DRS. Se toma como base el código fuente de la última versión aprobada del producto, y se realizan los cambios requeridos.
9. Actualización de diseño detallado: Se toma como base el diseño de la última versión del producto aprobada, y se actualiza con los cambios necesarios que den cuenta de los cambios realizados según sea necesario.
10. Verificación y Validación:
    Pruebas de regresión: Es el proceso de pruebas que lleva a cabo QA para asegurar que los cambios realizados a algunos componentes del sistema no afectan a otros componentes no modificados. Este proceso puede incluir la repetición de los casos de prueba de la última versión del sistema y que estén directamente relacionados con la parte del sistema que se ha modificado.
    Pruebas de aceptación: Se divide en pruebas OFAT y OSAT, y se aplica de la misma manera que en el proceso de desarrollo normal.
11. Entrega (transferencia) y configuración de software: Se aplica de la misma manera que en el proceso de desarrollo. Una vez aceptado en producto en OSAT, se actualizan lo ítems de configuración y se dejan en control de versiones, actualizando el Manual de operación y mantención de software.

El flujo de estás actividades se organiza de la siguiente manera.

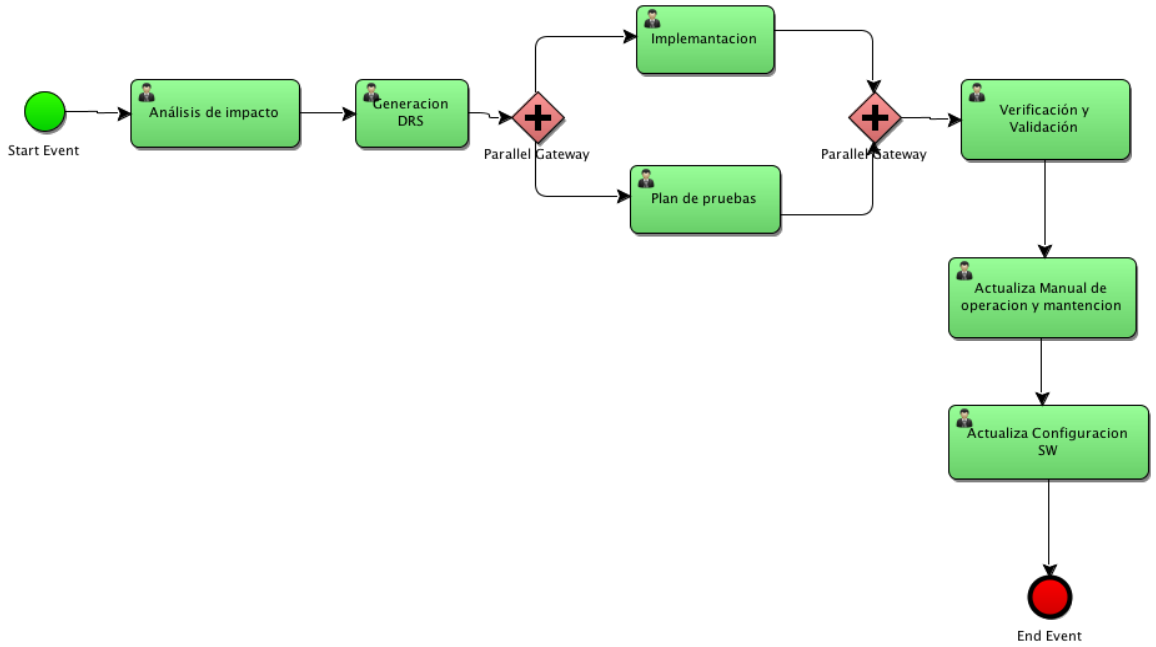


Figura 20: Proceso para cambios al software

El flujo se inicia con la llegada de una solicitud de cambio de software, indicando:

a. Cambios funcionales a realizar
b. Producto de software afectado
c. Detalle de los cambios solicitados

Donde:

1. El jefe de mantención realiza un análisis de impacto, considerando:
    a. Sistemas afectados
    b. Recursos involucrados
    c. Riesgos detectados
    d. Aprobación por el comité de cambios.
2. El ingeniero de mantención actualiza arquitectura, y confecciona el DRS.
3. El ingeniero de mantención realiza la implementación de los cambios, y actualizando el diseño detallado de la solución, y pruebas automáticas.
4. El ingeniero de QA actualiza planes de prueba, casos de prueba, y procedimientos de prueba según sea necesario.
5. El Ingeniero de QA realiza la validación y verificación del producto de software, considerando prueba de regresión, de OFAT y OSAT.
6. El ingeniero de mantención actualiza el manual de operación y mantención del producto.
7. El ingeniero de QA actualiza la configuración de software.

Para este proceso no se considera simulación debido a que no se tiene acceso a registros históricos de referencia.

4.1.3 Automatización de los flujos de proceso

4.1.3.1 Requerimientos

Desde el punto de vista del negocio y la mejora continua, la infraestructura base debe entregar las herramientas para mantener control sobre los procesos de manera eficiente, y aquellas que faciliten el cambio en los procesos como elemento principal del proceso de mejora.

Considerando lo anterior, la arquitectura definida deberá soportar el ciclo de vida de los procesos definido en BPM, de modo dar cumplimiento a los objetivos de mejorar la eficiencia en la gestión de los procesos.

A continuación los requerimientos que debe abarcar el sistema.

4.1.3.1.1 Requerimientos funcionales

- RF01 - Desplegar nuevo proceso: El sistema debe permitir agregar una nueva definición de proceso.
- RF02 - Iniciar Caso: El sistema debe permitir iniciar un nuevo caso de un proceso. Se debe permitir iniciar casos solo de la última versión de un proceso desplegado en el sistema.
- RF03 - Completar tarea pendiente: El sistema debe permitir ingresar la información correspondiente a la finalización de una tarea asignada, y asignar automáticamente la próxima tarea del proceso en cuestión.
- RF04 - Revisar estado de casos iniciados: El sistema debe permitir revisar el estado actual de los casos iniciados por el usuario actual del sistema.
- RF05 - Revisar estado de casos supervisados: El sistema debe permitir revisar el estado actual de los casos supervisados por el usuario actual del sistema.
- RF06 - Revisar estado de tareas completadas: El sistema debe permitir revisar el estado actual de las tareas completadas por el usuario actual del sistema.
- RF07 - Revisar tareas canceladas: El sistema debe permitir ver un listado de las tareas canceladas en el sistema.
- RF08 - Cancelar caso iniciado: El sistema debe permitir cancelar un caso iniciado por el usuario actual del sistema.
- RF09 - Cancelar caso supervisado: El sistema debe permitir cancelar un caso supervisado por el usuario actual del sistema.
- RF10 - Ver estado de procesos: El sistema debe permitir ver el estado actual de los procesos del sistema.
- RF11 – Diseñar nuevo proceso: El sistema debe proveer las facilidades para diseñar o cambiar el flujo de un proceso.

Los casos de uso de estos requerimientos se encuentra definidos en el anexo H.

4.1.3.1.2 Requerimientos no funcionales

Requerimientos de interfaz de usuario

- El acceso al sistema se hace a través de intranet, y el usuario interactúa con el sistema a través de un navegador web.

- El usuario verá solo aquellas opciones que su perfil le permiten acceder.

Requerimientos de seguridad

- Los usuarios deberán identificarse para poder acceder al sistema.

- La información de acceso debe ser consultada a un servidor LDAP provisto por el departamento de sistemas.

4.1.3.1.3 Restricciones de implementación

El software debe ajustarse a las restricciones de sistemas operativos, y bases de datos definidos como estándar dentro de la organización.

Estos son:

- Sistema operativo: Ubuntu 8.04 server
- Lenguaje de programación: Java SE 6
- Servidor de aplicaciones: Glassfish App Server 3.0 o superior
- Base de datos: PostgreSQL 8.0 o superior

4.1.3.2 Definición de la arquitectura de la solución

Siguiendo los principios de BPM, la solución debe respetar la separación entre la definición de los procesos y la ejecución de los mismos, lo que facilita y agiliza el cambio en los procesos.
Cómo principales características de la plataforma base de la solución se consideran:

- Herramienta de diseño de procesos.
- Administración de procesos.
- Monitoreo de procesos.
- Despliegue de los procesos
- Ejecución de los procesos.
- Interfaces para la provisión de usuarios desde un servidor LDAP.
- Definición de formularios para el despliegue.

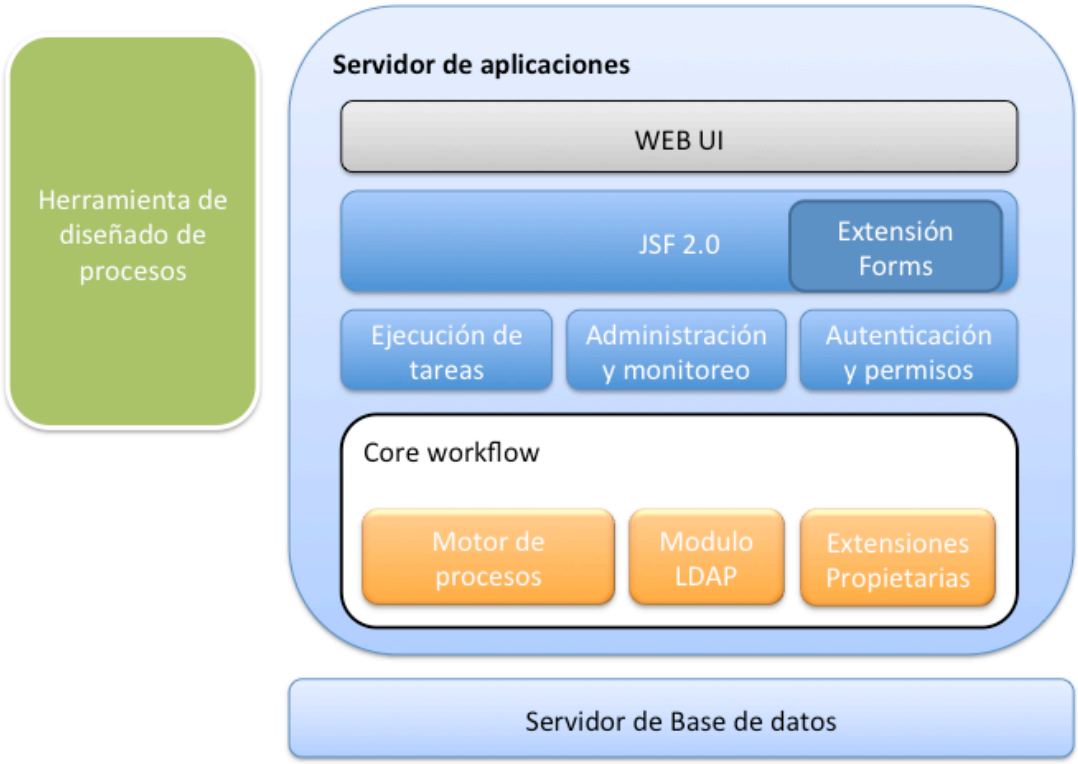


Figura 21: Arquitectura de la solución

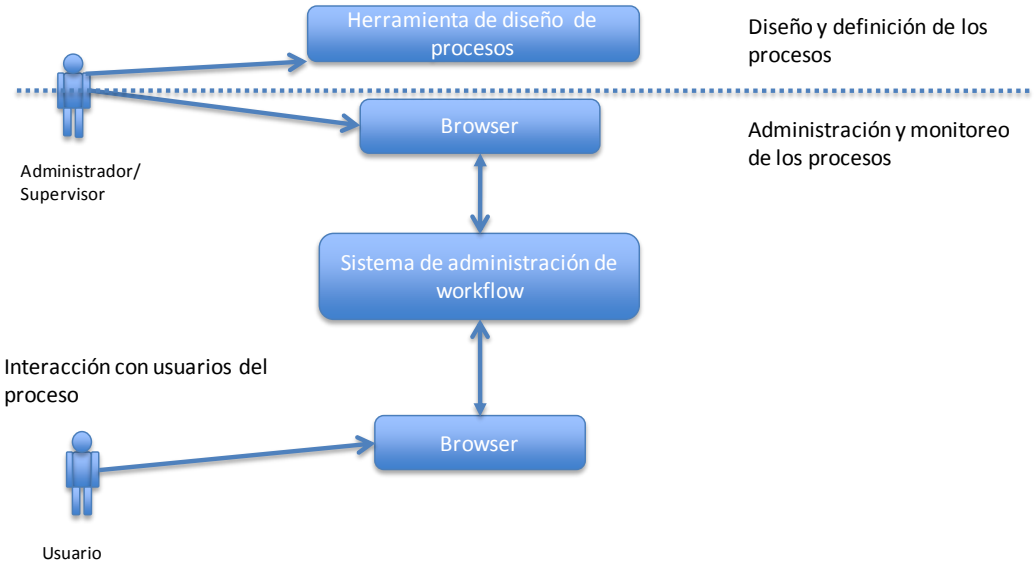


Figura 22: Responsabilidades y uso del sistema

4.1.3.2.1 Opciones para la implementación

Para la automatización de los flujos de trabajo se han considerado las siguientes alternativas:

Activiti

Activiti [12] es un framework de un motor de procesos BPMN 2.0 que implementa la especificación BPMN 2.0. Sobre su motor de procesos se puede desplegar definiciones de procesos, iniciar una nueva instancia de procesos y ejecutar tareas de usuario, entre otras funcionalidades contempladas por BPMN 2.0.

jBpm

jBPM [28] es un motor de procesos open source, el que en principio soportó una versión adaptada del lenguaje de procesos jPDL. A partir de la versión 5.0 del software se agregó soporte para BPMN 2.0.

Joget

Joget [29] es un sistema de administración de workflows (WFMS) que sirve como plataforma para que usuarios finales puedan diseñar, desplegar y ejecutar workflows para sus procesos de negocios.

En base a un conjunto de características relevantes para el trabajo se hizo una tabla comparativa:

	Activiti	jBPM	Joget
Capacidad de ejecutar tareas de usuario.	SI	SI	SI
Capacidad de ejecutar tareas de sistemas, de modo de poder integrar otro software dentro del workflow.	SI	SI	NO
Compatible con estándar bpmn 2.0.	SI	SI	NO
Comunidad activa, y equipo de desarrollo que la soporta	SI	SI	NO
Facilidad de uso	SI	SI	SI
Facilidad de adaptación	SI	SI	SI

Tabla 13: Evaluación motor de BPM

Tanto Activiti como jBPM cumplen en algún grado con las características deseadas del software base.

Finalmente se optó por Activiti dada la experiencia del equipo de desarrollo, el cual esta liderado por los desarrolladores del núcleo de jBPM, y la fuerza de la comunidad que lo soporta, en la cual se incluyen Alfresco, SpringSource, FuseSource y Mulesoft.

Dentro de las facilidades de adaptación se encontró de suma utilidad la abstracción que hace de la provisión de usuarios utilizados en la asignación de tareas. Esto permite que podamos reemplazar el uso del repositorio local de Activiti por una interfaz que conecte a repositorios de otro tipo de manera casi transparente para el motor de workflow.

Además de lo anterior cabe destacar que adicionalmente al soporte de BPMN2.0, Activiti incluye extensiones que permiten incluir dentro del flujo definido tareas de mensajería a buses de integración de sistemas como ESB, o a componentes propias que hagan de interfaz con sistemas externos.

4.1.3.2.2 Requerimientos de la implementación (Hardware y Software)

Para la implementación de la arquitectura señalada en el punto anterior se consideran los siguientes requerimientos.

Servidor

- 2 Procesadores dual core.
- 2 GB de memoria.
- 10 GB mínimo para almacenamiento y base de datos
- Ubuntu 8.04 server
- Java EE 6
- Glassfish App Server 3.0 o superior
- PostgreSQL 8.0 o superior

Estación de trabajo

- Procesador Pentium IV
- 512 MB de memoria
- Linux o Windows XP
- Conexión a internet o al servidor de procesos dentro de la red local
- Internet Explorer 7, Mozilla FireFox o Google Chrome.

4.1.3.2.3 Breve descripción del sistema de flujo y control de procesos desarrollado

El sistema para la automatización y gestión del flujo de los procesos definidos en 4.1.1 y 4.1.2, ha quedado como sigue:

- **Herramienta de diseño de procesos**: Yaoqiang BPMN, herramienta de diseño compatible con Activiti, con explorador LDAP incorporado y extensible a través de plugins. A esta herramienta se le potencia mediante la implementación de un plugin que permite el despliegue directo desde la herramienta al motor de procesos(ver anexo D).

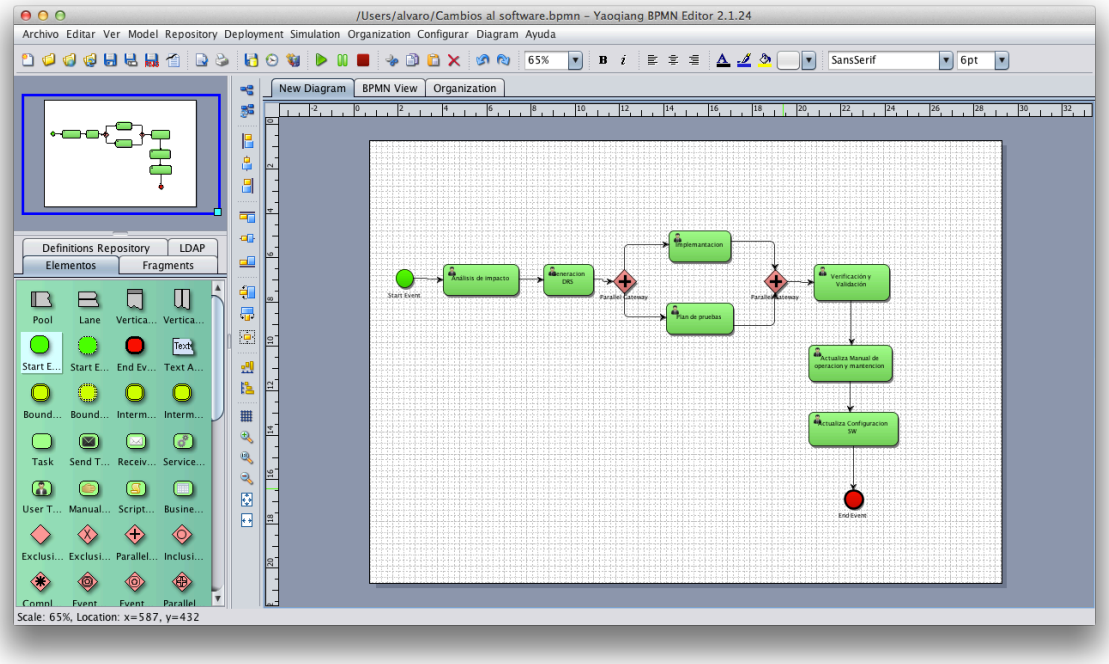


Figura 23: Herramienta de diseño BPMN Yaoqiang

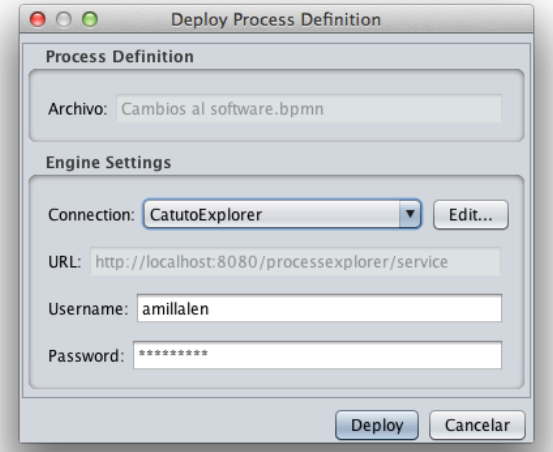


Figura 24: Despliegue de modelos sobre el motor de BPM

- **Simulador de procesos**: BizAgi v2.5.1.1, Herramienta que puede ser utilizada en las primeras propuestas de proceso para tener una aproximación a su desempeño para un escenario de carga proyectada.

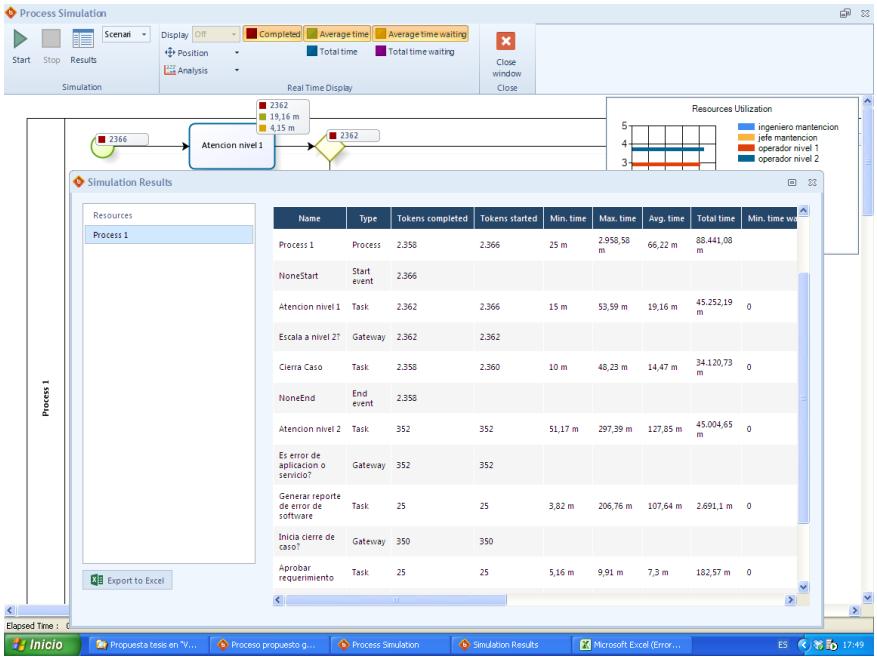


Figura 25: BizAgi como simulador de procesos

- **Explorador de procesos (Catuto):** Herramienta construida sobre el motor de bpm Activiti que permite desplegar procesos diseñados con Activiti designer, iniciar instancias de proceso, hacer seguimiento de la actividad y monitorear el rendimiento de los procesos a través de indicadores generados a través de esta misma herramienta(ver anexo C, y anexo E).

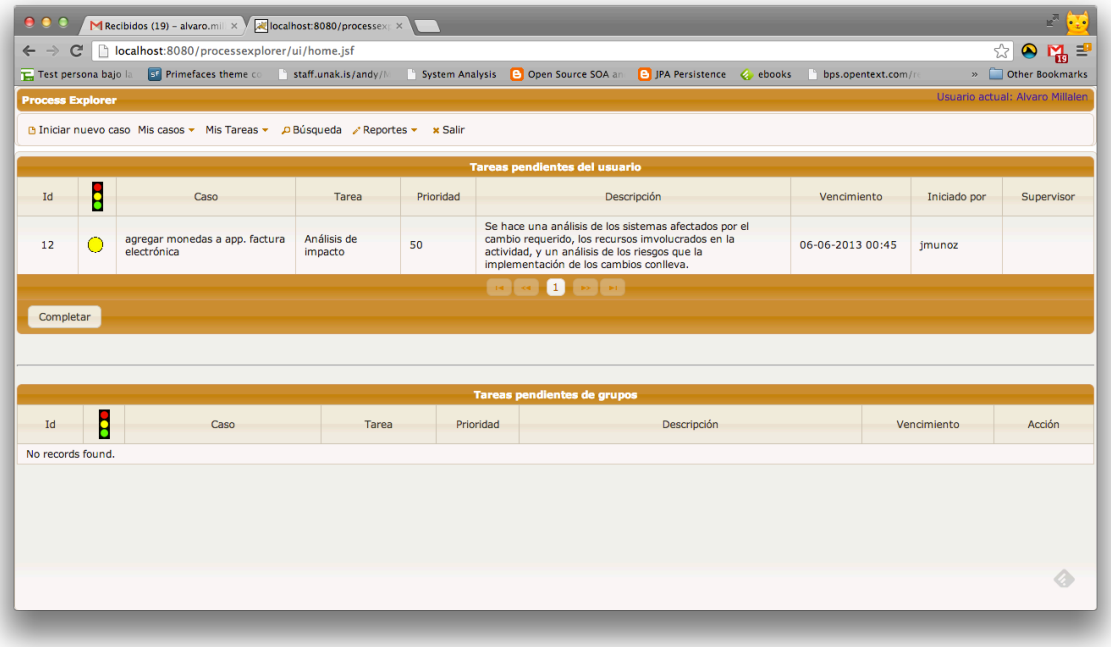


Figura 26: Explorador de procesos Catuto

- **Administración de usuarios**: Se integra con servidor LDAP existente de modo de simplificar la administración de usuarios (para más información sobre la integración con LDAP ver anexo C).

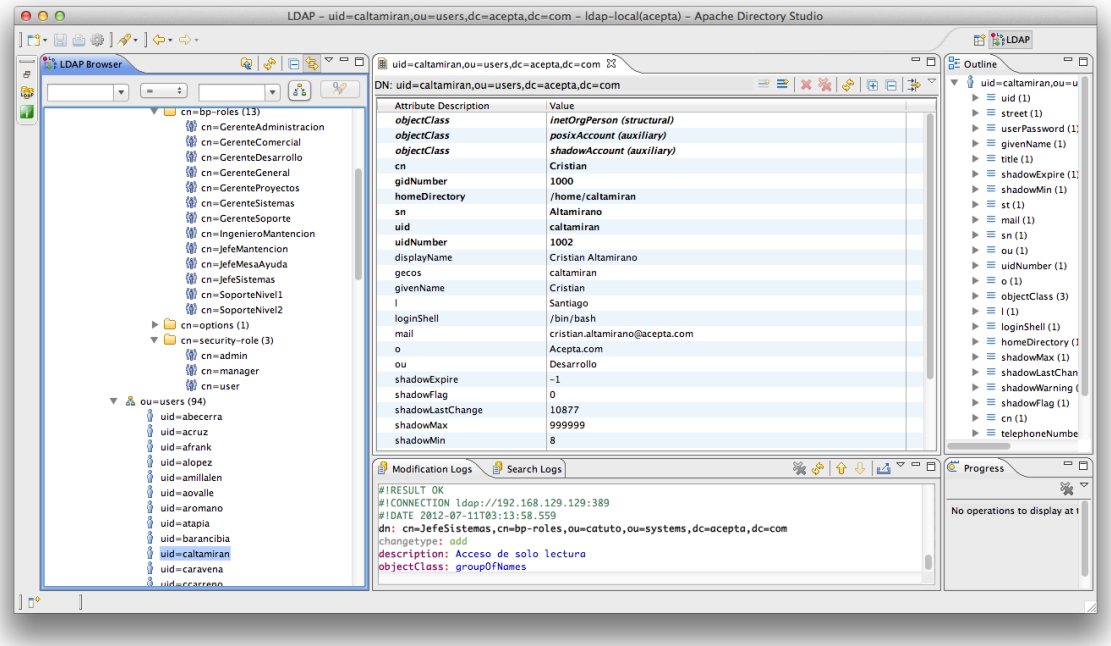


Figura 27: Administración de usuarios con LDAP

- **Definición de formularios del proceso**: En la actual etapa esta tarea puede ser realizada a través de la herramienta de diseño de procesos para los casos más simples. Para los casos avanzados esta tarea debe ser asistida por un

desarrollador, pues los formularios avanzados están desarrollados sobre tecnología JSF2.0.

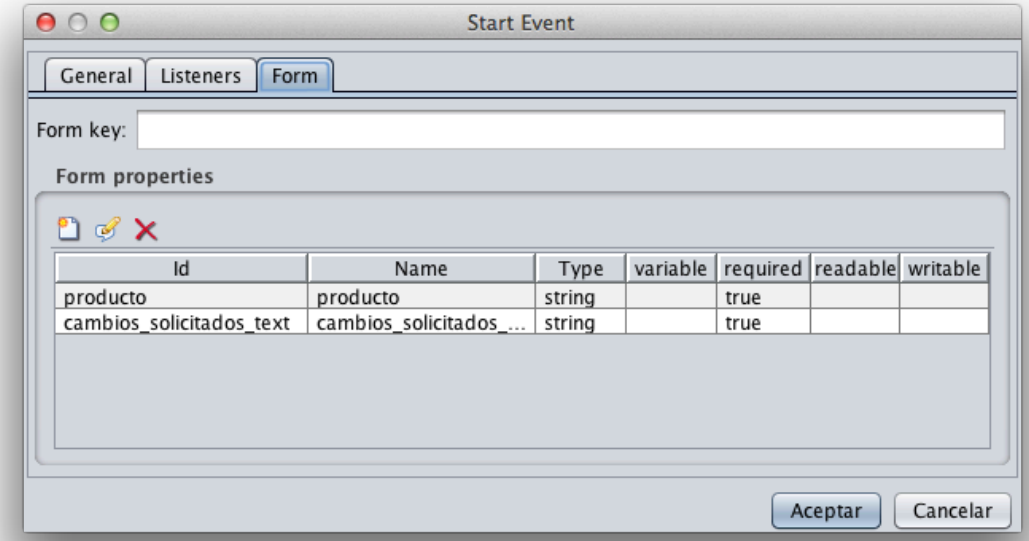


Figura 28: Definición de formularios

- **KPI de procesos:** Para la medición y evaluación del estado de los procesos se definieron tres indicadores, los que se encuentran incorporados como parte del software explorador de procesos (ver anexo C, monitoreo de desempeño de los procesos).

4.2 DEFINICIÓN DE LA ORGANIZACÓN DE MANTENCIÓN, MEJORAMIENTO DE PROCESOS Y PRÁCTICAS

4.2.1 Organización de mantención

La organización de mantención dependerá directamente del gerente de desarrollo. Estará encabezado por un jefe de mantenimiento, el que tendrá como recursos para su tarea a ingenieros de software, quienes llevarán a cabo las tareas propias de la mantención correctiva y mantención por solicitudes de cambio(adaptativa, correctiva, y perfectiva).
En este esquema los recursos de QA, son recursos compartidos con el área de desarrollo de software.

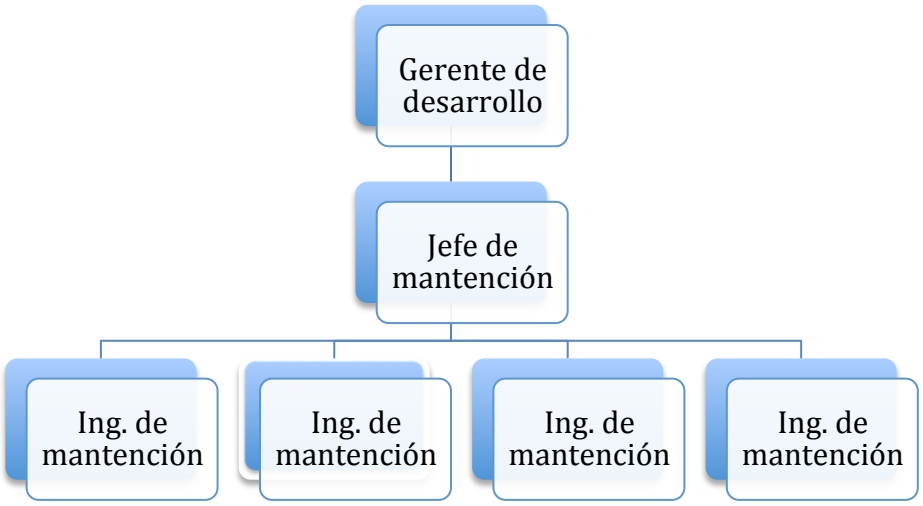


Figura 29: Estructura organización de mantención

Se establecen además las siguientes políticas para el mantenimiento del software:

- Las actividades de mantención de software correctiva tendrán como finalidad siempre tener el software operativo y corregido en el menor tiempo posible, cuidando la calidad y continuidad del servicio.
- Las especificaciones funcionales del software son conocidas por todos los participantes en la cadena de gestión de problemas, y son de fácil acceso para el personal involucrado en estas actividades.
- Todo cambio funcional del software debe ser autorizado por los niveles que correspondan (comité de producto, gerencia de proyectos, gerencia de desarrollo), y comunicados una vez liberados.
- Todo nuevo software debe considerar pruebas automáticas, como parte de los entregables, principalmente para las funciones criticas del sistema.

4.2.2 Herramientas de software para la mantención de los sistemas

Las herramientas a utilizar para la gestión y la operación de los procesos de mantención de software son:

- Java Decompiler: Herramienta utilizada en aquellos casos en que se enfrenten problemas con sistemas legados de los cuales no se tenga un control adecuado de las fuentes. Permite descompilar componentes y realizar análisis, y reconstruir las fuentes de en caso de perdida de estos.

- Eclipse IDE: Como entorno para la edición y compilación de las componentes o sistemas a modificar.

- AmaterasUML: Conjunto de plugins para el IDE eclipse que apoyan la tarea de modernización de sistemas legados o la mantención evolutiva de los sistemas una vez en producción. De este software la función más valiosa para nuestro proceso de mantención es la función de ingeniería reversa de los módulos, y la generación automática de los modelos que los representan.

- Herramienta de workflow de procesos: Herramienta de workflow para la gestión y manejo de las tareas del proceso de gestión de problemas y mantención correctiva.

- Eclipse mylyn: Plugin para eclipse para el manejo de las tareas administrativas. A través de este sistema los mantenedores tienen acceso a la lista de trabajos que tienen pendientes en la herramienta de workflow de procesos.

4.2.3 Proceso de desarrollo actual

El proceso de desarrollo de la organización se basa en las prácticas y procedimientos sugeridos por el estándar ESA para proyectos pequeños definidos en “Guide to applying the ESA software engineering standards to small software projects, BSSC(96)2 Issue 1.” [20]. No se hace mención a una organización de mantención, ni sus procesos.

En resumen, las actividades del ciclo de desarrollo establecidas en el proceso actual son las siguientes:

1. Diseño de Arquitectura, Diseño básico de la solución que incluye cierta cantidad de diagramas (de bloques, de secuencia, etc.) y flujos que explican el funcionamiento de la solución que se entregará. Este diseño se incluye en el DRS.
2. Plazos comprometidos en el desarrollo. En el DRS se incluyen además los plazos comprometidos en la codificación de lo especificado en el documento.
3. Plan de pruebas de aceptación. Este documento se genera una vez conocido el DRS y especifica a grandes rasgos, los aspectos relevantes a

probar en el sistema de tal forma de incluir comprobaciones para cada uno de los requerimientos de usuario existentes.

4. Casos y procedimientos de prueba. Definido y aprobado el plan de pruebas, se puede trabajar en los casos y procedimientos de prueba de tal forma de asegurar que las pruebas sean lo suficientemente completas que abarquen y permitan demostrar que se satisfacen cada uno de los requerimientos de usuario. Usualmente esta fase se trabaja en conjunto con el cliente para llegar a acuerdo de lo que se va a probar y obtener de él, el conjunto de datos y procedimientos de prueba que se utilizará.
5. Codificación, Es el proceso de construcción del software solicitado en el DRU y especificado en el DRS.
6. Diseño Detallado, Diseño completo y detallado de la solución, tal cual se entregó luego de la codificación. Este documento es parte de la documentación final del proyecto y contiene diagrama de deployment, diagramas actualizados de bloques, flujos, secuencias, clases, etc. (según apliquen).
7. Verificación y Validación.
    a. Pruebas de Integración, Es el proceso de pruebas internas que aplica QA integrando la solución codificada con todos los componentes que lo incluyen.
    b. Pruebas de Sistema, es el proceso de pruebas internas que aplica QA y que son el primer paso en la aceptación del desarrollo.
    c. Pruebas de Aceptación. Se incluyen dos actividades:
        i. Pruebas de aceptación “en fábrica” (OFAT) que son pruebas formales de QA donde se recorre el plan de pruebas completo y ejecuta un subconjunto representativo de casos de pruebas que permitan demostrar que lo programado satisface lo solicitado.
        ii. Pruebas de aceptación en “el sitio” (OSAT) que son pruebas de aceptación que se ejecutan con el cliente donde se pone en marcha el plan de pruebas, los casos, procedimientos, escenarios y datos de prueba y se demuestra el cumplimiento cabal de toda la aplicación desarrollada con respecto al DRU.
8. Entrega (transferencia) y configuración del software. Aceptado el desarrollo (OSAT) este pasa al ambiente de producción (ya sea en el cliente externo o en las instalaciones de la empresa). La versión en producción se marca como tal en el sistema de control de versiones y se almacena el código fuente, configuraciones y documentación de tal manera de poder identificar la versión entregada y asegurar su integridad.

4.2.4 Mejoras a la fase de transferencia

Como parte de las modificaciones al proceso de desarrollo, se genera una interfaz entre este proceso y operación y mantención de sistemas.

Para esta interfaz se incorpora un “Manual de Operación y Mantención del Sistema”. Este documento será un entregable de la fase de transferencia definida actualmente, y en él se especifican:

1. Propósito del sistema
2. Características principales del software
3. Ambiente necesario para el correcto funcionamiento del software(hardware, software base, actividades operacionales)
4. Operación del sistema
5. Arquitectura del sistema
6. Instalación del sistema
7. Uso del sistema
8. Administración del sistema
    a. Administración del cambio
    b. Administración de la configuración
    c. Administración de release
    d. Administración de seguridad
    e. Administración del sistema
9. Administración del servicio
10. Requerimientos regulatorios.

En este documento se definen cada uno de los puntos o se hace referencia al documento que lo define en el contexto del proyecto o definiciones de nivel empresa. El objetivo del Manual de operación y mantención es permitir la transferencia desde el equipo de proyecto a los equipos de operación y mantención del sistema, facilitando su operación y mantención durante su vida útil.

Cómo buena práctica se establece una plantilla de Manual de Operación y Mantención a partir del cual se generan/actualizan los documentos para cada proyecto (ver anexo I).

4.2.5 Mejoras a la Calidad Intrínseca

Para mejorar la calidad intrínseca de los productos de software fabricados, y mantenidos por la empresa, se introdujeron las siguientes prácticas:

1. Producción de tests automáticos como una práctica habitual durante toda la etapa de desarrollo. Esto ayuda a la detección temprana de fallas introducidas al sistema durante el proceso de mantenimiento, mejorando la mantenibilidad de los sistemas y ayudando al traspaso del conocimiento, al utilizarlo como documentación del comportamiento del sistema (ver 2.1.7).
2. Generación y administración de un repositorio central de librerías. Esto ayuda a la reducción de redundancia de código en los diferentes proyectos, y favorece la reutilización de los artefactos de software producidos.
3. Utilización de componentes en el desarrollo web. Esto permite agilizar el desarrollo, ya que el 80% de las necesidades de la interfaz web ya están cubiertos por varias frameworks, ya sea de libre uso o pagadas. De esta forma se favorece el concentrar los recursos en las capas de negocio de las aplicaciones.

Las herramientas utilizadas para materializar estas prácticas fueron:

Herramienta	Descripción
JUnit	Framework para la programación de tests automáticos [45].
Apache maven	Herramienta para la gestión y construcción de proyectos Java. Puede manejar todo el ciclo de construcción de los artefactos del proyecto, reportes, documentación, y otros dependiendo de los plugins que se configuren [46].
Sonatype Nexus	Implementación de un repositorio de artefactos maven, desde el cual se puede consultar, descargar, y publicar, automáticamente librerías [47].
Primefaces	Framework de desarrollo web por componentes basado en el estándar JSF [48].
Eclipse m2e	Plugin para la herramienta de desarrollo eclipse. Permite utilizar maven directamente desde el editor de código [49].
Eclipse	Herramienta de para el desarrollo de software [50]. Es el estándar utilizado por la empresa como interfaz de desarrollo y mantención de software.
Subversion	Software para el control de versiones de código [51]. Es el estándar utilizado en la empresa para el control de versiones de los proyectos.

Es importante hacer notar la importancia del uso de maven para la gestión de la construcción de los artefactos de software, ya que la herramienta facilita la construcción de un estándar en la estructura del software, y guía la aplicación de tests automáticos. Permitiendo la generación de los releases de sistema, sólo en caso de que los tests sean exitosos.

Los proyectos basados en maven, consideran fases consecutivas por las que se debe pasar para construir el software (ver Figura 30). Como consecuencia de esto, maven generará los paquetes de software sólo si no se detectan fallas al correr los tests automáticos.

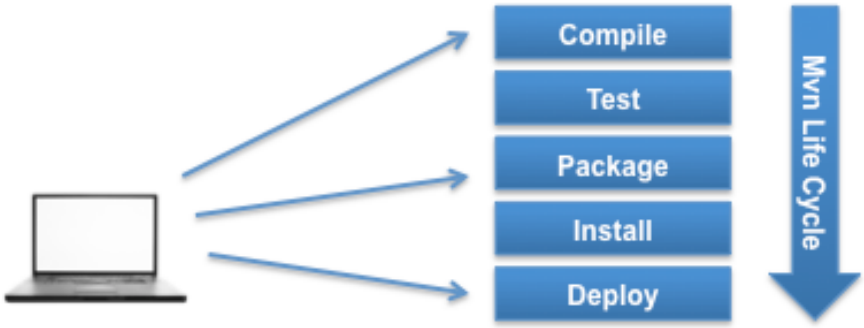


Figura 30: Fases de construcción de un proyecto maven

El stack de herramientas para el desarrollo de software integradas y su relación se puede apreciar en la Figura 31.

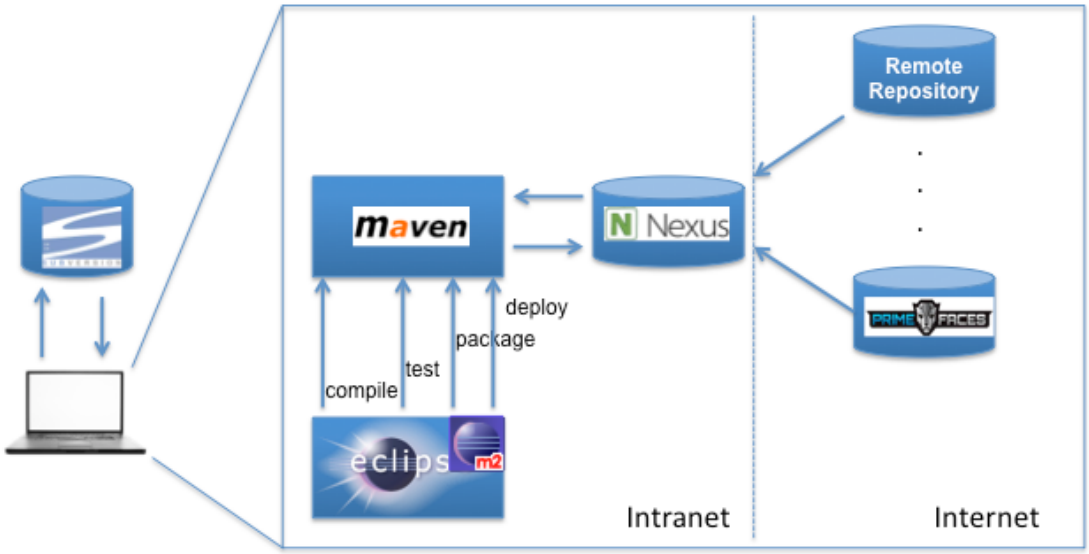


Figura 31: Herramientas de apoyo a la calidad del software

4.2.5.1 Métricas del código generado

Un complemento a la incorporación de la práctica de tests automáticos es la incorporación de la medida de cobertura de pruebas de los sistemas. Para la generación de los informes de cobertura se considera el uso del plugin para maven “Cobertura” [52], el que genera un informe detallado de métricas del proyecto indicando % de cobertura de las pruebas, y complejidad, de las clases y módulos considerados. Como ejemplo existen dos proyectos, en ellos, se construyeron tests automáticos para 146 clases en total, cuyos informes de métricas de cobertura se pueden consultar en el anexo B.

4.3 COSTOS ASOCIADOS A LA IMPLEMENTACIÓN

A continuación, se muestran los principales costos asociados a los ítems de mejoras propuestas. Lo anterior basado en los datos obtenidos de la simulación del proceso, y su configuración resultante en termino de cantidad y uso de los recursos humanos (ver 4.1.1.3.3).

Se han considerado los costos de la solución mínima que permite cumplir con los objetivos de mejora, quedando los costos de la siguiente manera:

Costo inicial: Aquí se han considerados los costos de desarrollo inicial (Tabla 14), los costos el costo del hardware y software necesario (Tabla 15), y los costos de instalación y puesta en funcionamiento del servidor utilizado (Tabla 16).

Costos de mantención: Aquí se han considerado los costos de mantención evolutiva del software de soporte a procesos establecido en la propuesta de mejora (Tabla 17).

Costos de operación: Aquí se han considerado los costos de operación de la solución durante tres periodos anuales. Se considera el caso en que se utiliza el sistema de tickets y procesos actuales, y el caso en que se modifica el proceso y se automatiza los flujos (Tabla 18 y Tabla 19).

Rol	% Uso	Tiempo(meses)	Costo empresa(mensual)	Costo proyecto
Jefe proyecto	0.2	3	2.040.000	1.224.000
Ing. Desarrollo	0.5	2	1.560.000	1.560.000
Analista Proceso	0.2	3	1.560.000	936.000
			Costo total en RRHH	**3.720.000**

Tabla 14:Costos de desarrollo de infraestructura para automatización de procesos inicial(inversión)

Ítem	Costo
Hardware	2.500.000
Software	0
Costo total	**2.500.000**

Tabla 15:Costo de infraestructura computacional necesaria para automatización de procesos

Rol	Tiempo(horas)	Costo empresa(mensual)	Costo proyecto
Jefe de proyectos	2	2.040.000	22.666,66667
Administrador de sistemas	7	1.470.000	57.166,66667
		Costo total	**79.833,33333**

Tabla 16:Costo de instalación de infraestructura computacional y software de soporte a procesos

Rol	Año 1	Año 2	Año 3
Jefe de proyecto	500.000	250.000	250.000
Ing. de mantención	2.808.000	1.404.000	702.000
Costo total	**3.308.000**	**1.654.000**	**952.000**

Tabla 17:Costos de mantención evolutiva de la infraestructura para procesos

Ítem	Año 1	Año 2	Año 3
Analista de proceso	3.744.000	3.744.000	3.744.000
Capacitación(relator)	156.000	156.000	156.000
Mantención del hardware	1.000.000	1.000.000	1.000.000
Mantención del software base(BD, backups)	180.000	180.000	180.000
Gestión de reclamos	4.279.502	4.279.502	4.279.502
Costo total	**9.359.502**	**9.359.502**	**9.359.502**

Tabla 18:Costos de operación proceso propuesto

Ítem	Año 1	Año 2	Año 3
Analista de proceso/Coordinador Mesa de ayuda	3.744.000	3.744.000	3.744.000
Capacitación(relator)	156.000	156.000	156.000
Mantención del hardware	1.000.000	1.000.000	1.000.000
Mantención del software base(BD, backups)	180.000	180.000	180.000
Gestión de reclamos	9.117.126	9.117.126	9.117.126
Costo total	**14.197.126**	**14.197.126**	**14.197.126**

Tabla 19:Costos de operación proceso actual

Ítem	Año 1	Año 2	Año 3
Costo Inicial	3.799.833,333		
Costo de mantención infraestructura de procesos	3.308.000	1.654.000	952.000
Costo de operación	9.359.502	9.359.502	9.359.502
Costo total	**16.467.335,33**	**11.013.502**	**10.311.502**

Tabla 20: Resumen de costos mejora propuesta

Ítem	Año 1	Año 2	Año 3
Costo Inicial	3.799.833,333		
Costo de mantención infraestructura de procesos			
Costo de operación	14.197.126	14.197.126	14.197.126
Costo total	**17.996.959,33**	**14.197.126**	**14.197.126**

Tabla 21: Resumen de costos proceso actual

CAPÍTULO 5 ANÁLISIS Y DISCUSIÓN DE LAS SOLUCIONES DESARROLLADAS

5.1 ESTADO ACTUAL DE IMPLEMENTACIÓN

Se ha formalizado la organización de mantención teniendo dos personas dedicadas a tiempo completo a esta labor y otras dos personas a tiempo parcial.
Los cambios y prácticas propuestas para el proceso de desarrollo, y el de mantención de software han sido introducidos en dos proyectos uno actualmente en ejecución (ACM-Administrador de Correo Masivo), y otro en etapa de mantención (Servicio de firma electrónica centralizada).

En ambos casos la incorporación del manual de operación y mantención de software, colaboró e hizo mucho más fácil el proceso de paso a producción de las aplicaciones. Es así como el proceso hoy no toma más de treinta minutos. Anteriormente los pasos a producción se caracterizaban por una iteración constante entre el operador de sistemas y el desarrollador a cargo. Estas iteraciones tenían por fin, resolver las configuraciones e integración con el resto de los sistemas, tomando en general entre dos y tres horas de tiempo.

	Antes de las mejoras al proceso	Después de las mejoras al proceso
Tiempo paso a producción	2 a 3 horas	30 minutos

Tabla 22: Mejoras observadas en los pasos a producción

Además se observaron las ventajas de la incorporación de tests unitarios automáticos en el proyecto de firma electrónica centralizada. En este caso, después de un mes en producción se necesitó hacer modificaciones correctivas para uno de los reportes. Los tests automáticos, previamente creados durante la etapa de desarrollo, ayudaron a verificar rápidamente el correcto funcionamiento de las otras funcionalidades relacionadas, en no más de treinta segundos por iteración. Este proceso anteriormente significaba el *deployment* en un servidor y necesariamente la verificación humana de todas las funcionalidades, lo que implicaba entre cinco a quince minutos por cada iteración de pruebas (Tabla 23).

	Antes de incorporación de pruebas unitarias	Después de incorporación de pruebas unitarias
Tiempo pruebas mantención correctiva	15 minutos	30 segundos

Tabla 23: Mejoras observadas en mantención correctiva

Lo anterior contrasta con la realidad de otros proyectos de la empresa, anteriores a la incorporación de estas modificaciones. Un ejemplo de esto es el sistema de cuadratura. Este sistema a lo largo de sus ocho años de existencia, ha sido objeto de modificaciones tanto correctivas y evolutivas. En todos las ocasiones se tuvo que:

- Competir por los recursos asignados a proyectos de desarrollo de nuevos sistemas.
- Enfrentar el poco conocimiento documentado de su evolución e interacción con los sistemas internos.
- Realizar una tarea de Análisis de problema/modificación que se extendió por días y hasta semanas en la recolección de información.

Hubo un incidente en el cual se degradó el rendimiento del sistema debido a la no mantención de unas tablas temporales. Esta mantención se llevaba a cabo por medio de un script ejecutado como tarea del sistema operativo. Lamentablemente en una migración de servidor se perdió, debido al desconocimiento de parte del operador del sistema de las dependencias de este. Este tipo de incidentes se evita en gran medida con la correcta implementación del manual de operación y mantención del software, que se propone en esta tesis.

Estas mejoras puntuales son de gran valor, sin embargo, queda por incorporar la automatización del flujo de los procesos de manera integral. Esta automatización permitirá agilizar y controlar de mejor manera la gestión de problemas y las correcciones de software asociadas. Actualmente la incorporación de la automatización recomendada, ha quedado en estado de espera, debido a reorganizaciones dentro de la empresa, y cambio de prioridades, por lo que se espera retomarlo a futuro.

5.2 DISCUSIÓN

En la presente tesis se buscó generar los cambios necesarios a los procesos y prácticas de la empresa para mejorar la eficiencia en la gestión de problemas y mantención de software. Para ello se siguió una metodología centrada en la mejora de procesos y en la automatización de sus flujos.
La validación de las soluciones desarrolladas fueron de dos tipos:

1. Validación por simulación. Esta se aplicó a la automatización de flujos del proceso de gestión de problemas y mantención de software. Para ello se validó utilizando datos históricos de la atención de clientes (ver 4.1.1.3). También se comprobó, mediante un análisis de estos datos, que la distribución de llegadas se aproximaba a la distribución exponencial, descrita en la bibliografía cómo la distribución recomendada a usar para este tipo de problemas [42]. Se comparó los resultados obtenidos con los esperados.
2. Validación por proyectos pilotos. Esta se aplicó a la creación de la organización de mantención, y cambios a los procesos de mantención y desarrollo. En este caso se piloteó un proyecto de mantención y otro de desarrollo. Sin embargo, en la mantención de software no hay datos históricos claros, por lo que se tuvo que recurrir a la opinión experta para obtener datos de comparación.

En la bibliografía relevante [21], se menciona las dificultades culturales que cabe esperar en la implementación de una organización de mantención. En nuestro caso, esto ocurrió, pero no fue una situación que no se pudiera manejar. Distinto es el caso de la implementación de los cambios a la gestión de los procesos transversales, como lo es el proceso de gestión de problemas. La explicación se encuentra en la reestructuración organizacional en la que se vio envuelta la empresa, y que no iba en la dirección de los cambios recomendados en la tesis. La empresa estaba cambiando de estrategia de negocios y paralizó la actuación de los ejecutivos. Otras áreas de la empresa comenzaron también a cuestionarse su futuro rol en la empresa.

Hasta antes de esta tesis la única métrica de producto mantenida era la de LOC (líneas de código). Sí se hubiera mantenido alguna métrica como las sugeridas en IEEE 1219 [22], ISO 9126 [26], y descritas en 2.1.3, se tendría una noción más clara de que tan mantenibles son los productos de software y se tendría una base para comparar y saber cual es el impacto de los cambios introducidos en la calidad del producto. En este sentido la introducción de los informes de cobertura de pruebas (ver anexo B) – desarrollados en esta tesis - son un avance importante. Estas metrícas si bien no son medidas de los procesos de desarrollo y mantención de software, sino de sus productos, son mucho más fáciles de recolectar, y la mejora de estas métricas tendrá algún impacto positivo en los tiempos asociados a la introducción de cambios al software.

5.3 DIFICULTADES ENCONTRADAS DURANTE LA IMPLEMENTACIÓN

Durante el desarrollo de esta tesis la empresa fue adquirida por un importante grupo con presencia principalmente en el área salud. Esto, como se mencionó en el apartado anterior, generó un proceso de cambios y reestructuración de la organización. Esto dificultó el avance de la implantación de la automatización de los flujos, impidiendo en primera instancia la implementación de un piloto, para finalmente quedar suspendido de manera indefinida. Para enfrentar esta dificultad se utilizó herramientas de simulación de procesos, que permitieran hacer pruebas del modelo y mejorarlo a través de simulaciones sucesivas.

También se experimentó las dificultades culturales asociadas con el trabajo de mantención de software, y creencia establecida de que es un trabajo de segunda categoría [21]. Esto dificultó el tener un equipo de trabajo estable para las tareas de mantención en esta etapa de implementación de las mejoras al proceso de mantención. Durante este tiempo se experimentó tres renuncias, de las cuales dos están asociadas al aspecto cultural del trabajo de mantención. Estas renuncias se suplieron con la contratación de una persona nueva para el equipo, y el traslado de dos ingenieros que realizaban tareas de desarrollo. Para enfrentar esta dificultad, al equipo de trabajo se le asignó también tareas de desarrollo (un proyecto). Esta estrategia funcionó, porque además los miembros del equipo debieron aprender herramientas nuevas, y eso redundó en motivación (véase 4.2.5).

La incorporación de tests unitarios automáticos, si bien tuvo buena recepción por parte del equipo, hubo dificultades para que los desarrolladores le dieran un uso práctico a la herramienta. En los primeros acercamientos siempre se dejaba la creación de los tests para cuando ya estaba escrito el código, y en otros casos los tests generados eran demasiado simples. En ninguno de los casos los test generados ayudaban al propósito de mejorar la calidad del producto, y comenzaron a verse como un problema, en vez de solución. Para enfrentar esta dificultad se implementó *pair programming*, entre quienes tenían mayor dificultad y aquellos que tenían más experiencia con la implementación de tests. También se implementaron minireuniones de cinco minutos, una o dos durante el día, para monitorear las dificultades encontradas, y evaluar soluciones.

El manual de operación y mantención - si bien tuvo buena acogida - tuvo resistencia por estar escrito en Word. Word, es una herramienta rechazada por el equipo de desarrollo. Una vez que se puso en práctica la generación del documento, la resistencia disminuyó debido a que el contenido generado fue el mínimo necesario, dejando en partes del documento referencias a otros documentos del SGC donde ya se habían definido los aspectos requeridos.

5.4 REVISIÓN DEL CUMPLIMIENTO DE OBJETIVOS

Como se explicó en 5.3, la verificación de los procesos respecto de los objetivos planteados se realizó mediante simulación. Los detalles sobre cómo se realizó la simulación se encuentran explicados en 4.1.1.3.
A continuación se presenta un breve análisis sobre el cumplimiento de objetivos.

Objetivo	Estado	Porcentaje estimado de cumplimiento
Reducir tiempo de respuesta de un requerimiento de cliente, de 102 horas observadas, a un máximo de 7 horas para gestión de problemas, y 72 horas para mantención correctiva de software (promedio).	Este objetivo se ha cumplido, en la medida que se han definido el tipo y cantidad de recursos, y los cambios de proceso necesarios para obtenerlo. Se ha validado su eficacia mediante los resultados obtenidos a través de herramientas de simulación de procesos. Se han definido completamente el proceso de mantención correctiva de software y su integración con el proceso de gestión de problemas. También se ha definido una organización de mantención de software para el adecuado aseguramiento de los recursos, y se han introducido mejoras en las prácticas del proceso de desarrollo que apoyen a la corrección. Estas mejoras se han puesto en práctica en dos proyectos de mantención y desarrollo.	100%
Reducir el tiempo necesario para implementar cambios de proceso de 4 días a 24 horas.	Este objetivo se ha cumplido, en la medida que se ha desarrollado una herramienta de automatización de procesos que permite la actualización del proceso en ejecución con el mismo diagrama bpmn utilizado hasta el momento como documentación.	100%
Gestionar el 100% de los casos con apoyo de TI, versus el 45% actualmente.	Este objetivo se ha cumplido, en la medida que se ha definido un proceso para la administración de la mantención correctiva. Este proceso se ha desplegado en la herramienta de automatización de flujos de procesos desarrollada, y queda a disposición de la organización.	100% (No se espera casos que sean procesados manualmente)

Tabla 24: Revisión de cumplimiento de objetivo

CAPÍTULO 6 CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

En el presente trabajo se ha analizado el proceso de gestión de problemas y mantención de software de la empresa, poniendo énfasis en el tiempo empleado en las tareas realizadas. Se han expuesto las debilidades de los procesos y prácticas de la empresa, como puntos de mejora importante, y se ha propuesto automatización de flujos de trabajo, cambios a los procesos, y la introducción de prácticas de desarrollo y mantención de software. Todos estos cambios se han introducido con la idea de mejorar el rendimiento del proceso completo, de modo de cumplir con los objetivos definidos en el punto 1.4 de este trabajo de tesis.

La solución de automatización del proceso de gestión de problemas y mantención de software comenzó con el levantamiento de los procesos existentes y sus problemas. Luego se diseño el cambio e integración del proceso de gestión de tickets y de cambio correctivo del software en BPMN. A continuación, a través de una herramienta de simulación de procesos se buscó la configuración de tareas y recursos que mejor cumpliera con los objetivos de mejora. Finalmente, se disponibilizó una herramienta de TI basada en un motor de flujos BPMN para el despliegue de los procesos definidos. La idea fundamental de esto último fue permitir a los responsables del diseño de procesos, introducir mejoras en diferentes iteraciones, haciendo el despliegue del diseño del proceso directamente en la herramienta de apoyo TI, y viendo resultados inmediatos.

La solución de definición de la organización de mantención de software, e introducción de cambios en las prácticas de desarrollo y mantención de software, comenzó con una definición de roles y responsabilidades de un equipo pequeño de mantención, continuó con mejorar la interfaz con el equipo de desarrollo y el de sistemas. Finalizó con la introducción de prácticas de pruebas automáticas, y desarrollo por componentes, basado en las ideas de TDD. La idea fundamental fue incorporar herramientas que permitan evolucionar los productos de software manteniendo el nivel de servicio entregado a los clientes.

Como producto secundario de este trabajo de mejora se encuentra la plataforma de ejecución y monitoreo de procesos. Esta plataforma permite realizar el ciclo definido por BPM de manera ágil, realizando el despliegue de las definiciones de procesos desde la herramienta de diseño de procesos. De este modo queda el proceso y su ejecución documentados en el mismo artefacto, lo que disminuye notoriamente la cantidad de documentación y agiliza la evolución los procesos.

Se utilizó en dos proyectos la interfaz definida entre el proceso de desarrollo y el de mantención de los sistemas, el cual fue de gran ayuda también como parte de la transferencia a los equipos que operan las soluciones de software en los ambientes de producción.

Se encuentra incorporada de manera estable en los nuevos desarrollos la práctica de producción de tests automáticos del software, los que han sido de gran ayuda para dar cumplimiento a las políticas del área de mantención de software.

El rediseño de los procesos utilizando BPMN 2.0, dada la experiencia de la empresa con BPMN 1.0, fue rápido y fácil de validar. Esto permite concluir que es un modelo que agiliza la manera de documentar y ejecutar los procesos, sin introducir cambios significativos en la manera en como se documentan los procesos hoy en la empresa.

La estructuración y definición clara de los roles y recursos disponibles, así como la definición de políticas e incorporación de buenas prácticas ha generado beneficios importantes al proceso de gestión de problemas y mantención de software. Los beneficios principales a la fecha son la disponibilidad de recursos para la actividad de mantención de software, y una mejora en los tiempos de respuesta asociados.

El apoyo computacional implementado ha demostrado ser una alternativa que permite el despliegue de los procesos diseñados en BPMN 2.0, acortando el ciclo de implementación de los procesos definidos con este estándar, y con el cual es posible dar cumplimiento al objetivo de reducir los tiempos de implementación de cambios a los procesos. A la fecha, este sistema no ha sido incorporado a la operación de la empresa, debido a una restructuración de la misma, y a los cambios de prioridades de los nuevos directivos.

Los cambios que se implementaron con mayor facilidad, fueron los relacionados con prácticas de desarrollo ágil, debido principalmente a que los involucrados pudieron ver resultados concretos en los productos de software que proveen los servicios en corto tiempo. En cambio aquellos cambios que involucraban al área de soporte y la introducción de una nueva herramienta para la administración de flujo de trabajo se vio suspendida, sin la posibilidad de mostrar resultados más allá de los obtenidos en la simulación.

La organización de mantención implementada y los cambios de proceso propuestos, permiten gestionar los problemas de clientes y realizar los cambios de software necesarios según los objetivos planteados al inicio de esta tesis. Esto basado en las simulaciones de procesos y el trabajo práctico realizado en el área de desarrollo y mantención de software.

6.2 RECOMENDACIONES

Se ha establecido una solución al problema planteado, implementando algunas de las prácticas regulares de la industria, y proponiendo la incorporación de tecnologías y prácticas de BPM para agilizar los cambios a través de herramientas de software disponibles para la empresa. Queda pendiente la incorporación a la operación de la herramienta de automatización de procesos desarrollada. Para ampliar los beneficios obtenidos se recomienda incorporar estas herramientas u otras similares, que permitan que la definición, documentación y puesta en funcionamiento de los procesos sean parte del mismo ciclo de mejora.

Es de mucha importancia aprovechar el conocimiento adquirido por la empresa con la implementación de ISO9001, fortaleciendo la práctica de la mejora continua de los procesos, para lo que se recomienda ampliamente la incorporación de tecnologías de BPM como base para la agilización de estos, en particular del proceso de gestión de problemas y mantención de software abordado en esta tesis.

El establecimiento e implementación de tests automáticos ha aumentado la facilidad de prueba de los productos. Es recomendable a futuro establecer la cobertura de pruebas como métrica de la mantenibilidad de los proyectos de software, y establecer políticas respecto del % de cobertura de clases, de *branches*, y complejidad del código enfocado en facilitar la evolución del software.

BIBLIOGRAFÍA

1. Weske , M., Goesmann , T., Holten , R., & Striemer , R. (2001). Analyzing, Modeling, and Improving Workflow Application Development Process. En *Journal on Software Process Management Improvement and Practice* (Vol. 6, págs. 35-46). Wiley.
2. Weske, M. (2000). *Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects.* Universidad de Münster.
3. *Wikipedia*. (s.f.). Recuperado el 2012, de http://en.wikipedia.org/wiki/Business_Process_Execution_Language
4. Wikipedia. (s.f.). *COCOMO*. Recuperado el 2012, de Wikipedia: http://es.wikipedia.org/wiki/COCOMO
5. Wikipedia. (s.f.). *Modelo Slim*. Recuperado el 2012, de http://es.wikipedia.org/wiki/Modelo_SLIM
6. Wikipedia. (s.f.). *Price systems*. Recuperado el 2012, de Wikipedia: http://en.wikipedia.org/wiki/PRICE_Systems
7. Workflow Management Coalition. (s.f.). *Workflow Reference Model*. Recuperado el 2012, de http://www.wfmc.org/standards/docs/tc003v11.pdf
8. Van der Aalst, W. (2004). *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management.*
9. Van der Aalst, W., & van van Hee, K. (2004). *Workflow Management, models, methods and systems.* The MIT Press.
10. Van der Aalst, W., & Kumar, A. (2001). A Reference Model for Team-Enabled Workflow Management Systems. En *Journal Data & Knowledge Engineering* (Vol. 38).
11. Van der Aalst, W., ter Hofstede, A., & Weske, M. (2003). Business Process Management: A Survey. En *Business Process Management International Conference.* Springer.
12. Activiti.org. (s.f.). *Activiti BPM Platform*. Recuperado el 2012, de http://www.activiti.org/
13. April, A., Huffman Hayes , J., Abran , A., & Dumke , R. (2004). Software Maintenance Maturity Model (SM3):The software maintenance process model. En *Journal of software maintenance and evolution.* Wiley.
14. Bajaj, A., & Ram, S. (2002). SEAM: A State-Entity-Activity-Model for a Well-Defined Development Methodology. En *IEEE Transactions on knowledge and data engineering* (Vol. 14). IEEE.
15. Baresi, L., Casati, F., Castano, S., Fugini, M., Mirbel, I., & Pernici, B. (1999). *WIDE Workflow Development Methodology.* ACM Digital Library.
16. Bizagi. (s.f.). *BizAgi homepage*. Recuperado el 2013, de http://www.bizagi.com/
17. Black, R., & Mitchell, J. L. (2011). Apéndice C: ISO 9126 Metrics. En *Advanced Software Testing-Vol. 3* (1a edición ed.).
18. Casati , F., Fugini, M. G., Mirbel , I., & Pernici , B. (2002). WIRES: A Methodology for Development Workflow Applications. En *Requirements Engineering* (Vol. 7, págs. 73-106). Springer.
19. CDC. (s.f.). *CDC project management templates*. Recuperado el 2013, de http://www2a.cdc.gov/cdcup/library/templates/default.htm
20. European space agency. (s.f.). *Guide to applying the ESA software engineering standards to small software projects*. Recuperado el 2012, de ftp://ftp.estec.esa.nl/pub/wm/anonymous/wme/bssc/Bssc962.pdf

21. IEEE Computer Society. (2004). Capítulo 6.Mantención de Software. En *Software Engineering Body of Knowdledge(SWEBOK).*
22. IEEE Computer Society. (1998). IEEE 1219-1998. IEEE Standard for software maintenance.
23. IEEE Computer Society. (1990). *IEEE 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.*
24. International Organization for Standarization (ISO). (2008). *ISO/IEC 14764. Software Engineering-Software Life Cycle Processes-Maintenance* (2a edición ed.).
25. International Organization for Standarization. (2008). *ISO/IEC 12207. System and software engineering-software life cycle processes* (2a edición ed.).
26. International Organization for Standarization. (1991). *ISO/IEC 9126. Information technology-software product evaluation-Quality characteristics and guidelines for their use* (1a edición ed.).
27. Hitpass Heyl, B. (s.f.). *El círculo mágico de la BPM-Suite.* Recuperado el 09 de 2012, de emb: http://www.emb.cl/gerencia/articulo.mvc?sec=7&num=547
28. jboss.org. (s.f.). *Jbpm Overview*. Recuperado el 2013, de http://www.jboss.org/jbpm/
29. joget.org. (s.f.). *Joget workflow*. Recuperado el 2013, de http://www.joget.org/
30. Kajko-Mattson, M. (2003). Capítulo 2, Corrective Maintenance within Problem Management. En M. Polo, M. Piattini, & F. Ruiz, *Advances in Software Maintenance Management: Technologies and Solutions.* Idea Group Publishing.
31. Kajko-Mattson, M. (2005). Capitulo 4:Upfront Corrective Maintenance at the Front-End Support Level. En *Managing corporate information systems evolution and maintenance.* Idea Group Publishing.
32. Khan , K., Lo, B., & Skramstad, T. (2001). Task and methods of software maintenance: a process oriented framework. ACS Digital Library.
33. Miers , D., & White, S. (2008). *BPMN Modeling and Reference Guide.* Future Strategies Inc.
34. OASIS. (s.f.). *Web Services Business Process Execution Language Version 2.0*. Obtenido de http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
35. Object Management Group. (2011). *Business Model Process and Notation v2.0*. Obtenido de http://www.omg.org/spec/BPMN/2.0
36. Pigoski , T. M. (1996). *Practical Software Maintenance. Best Practices for Managing Your Software Investment* (1a edición ed.). Wiley.
37. Pressman, R. S. (2005). *Ingenieria de Software, un enfoque práctico* (6a edición ed.). Mc Graw Hill.
38. Shi, M., Yang, G., Xiang, Y., & Wu, S. (1998). Workflow Management Systems: A survey. En *Proceedings of IEEE Intl Conf on Communication Technology.* Beijing.
39. Smith, H., & Fingar, P. (2003). *Business Process Management: The third wave.* Meghan-Kiffer Press.
40. Software Engineering Institute. (s.f.). *CMMI para Desarrollo, Versión 1.3*. Recuperado el 2012, de http://www.sei.cmu.edu/library/assets/whitepapers/Spanish%20Technical%20Report%20CMMI%20V%201%203.pdf
41. Sommerville, I. (2005). *Ingeniería de software* (7a edición ed.). Pearson.
42. Robinson, S. (2003). *Simulation: The Practice of Model Development and Use.* Wiley.

43. Álvarez Contreras, E. G. (2014). *Estudio comparativo de metodologías de desarrollo de software orientadas a la calidad intrinseca.* Universidad de Chile.
44. Demeyer, S., Ducasse, S., & Nierstrasz, O. (2013). *Object-Oriented Reengineering Patterns.* Square Bracket Associates.
45. JUnit.org. (s.f.). Recuperado el 2014, de JUnit: http://junit.org
46. Fundación Apache. (2014). Obtenido de Apache maven: http://maven.apache.org/
47. Sonatype. (2014). Obtenido de Sonatype Nexus: http://www.sonatype.org/nexus/
48. Primetek. (2014). Obtenido de Primefaces: http://primefaces.org/
49. Eclipse.org. (2014). Obtenido de Eclipse m2e: https://www.eclipse.org/m2e/
50. Eclipse.org. (2014). Obtenido de Eclipse: https://www.eclipse.org/
51. Fundación Apache. (2014). Obtenido de Apache subversion: https://subversion.apache.org/
52. (2014). Obtenido de Cobertura maven plugin: http://mojo.codehaus.org/cobertura-maven-plugin/

ANEXO A: ANTECEDENTES DE LA EMPRESA

Acepta.com es una empresa fundada en 1999 con la idea de desarrollar servicios de documentos electrónicos con respaldo legal, siendo su misión “Proveer al cliente soluciones digitales para mejorar la seguridad y eficiencia de sus procesos de negocio.”, y para lo cual genero dos líneas de negocio, documentación electrónica y certificación electrónica.

Sí bien tecnológicamente las soluciones desarrolladas cumplían con lo prometido, no fue sino hasta 2001, y dado el interés del mercado en el uso de documentos tributarios electrónicos que la empresa acercó posiciones con el Servicio de Impuestos Internos (SII), y le propuso un modelo operacional que permitía prescindir del papel como fuente primara de apoyo a la fiscalización, y disminuir la evasión de impuestos, y por lo tanto contó con un alto apoyo gubernamental. Por su parte Acepta.com coordinó a 5 grandes empresas en el uso de su modelo operacional de factura electrónica, cuyos resultados fueron presentados al SII, el cual inició su propio piloto de factura electrónica con 9 grandes empresas el año 2002. Acepta.com fue el proveedor tecnológico de 4 de las 9 empresas del piloto del SII. El 2003 el sistema fue abierto a todo el mercado y desde entonces Acepta.com ha mantenido una posición de liderazgo indiscutido en factura digital o electrónica. Este modelo actualmente es usado como referencia en varios países de Latinoamérica.

Paralelamente, se trabajó en conjunto con el Servicio de Registro Civil e Identificación de Chile, para permitir que los procesos de verificación de identidad, que conducen a la emisión de una cédula de identidad o pasaporte, también permitan la emisión de un certificado de firma electrónica. Este proyecto vio la luz el 2002 y posicionó a Chile como el primer país del mundo en que el Registro Civil se interconecta con un Prestador de Servicios de Certificación (PSC) Privado, logrando así los máximos niveles de confianza asociados a la entrega de certificados de clave pública (llamados Certificados de Firma Electrónica por la Ley Chilena).

Acepta.com colaboró en Chile con el Poder Ejecutivo y Legislativo en la elaboración de la Ley de Firma Electrónica (incluyendo su reglamento y guías de acreditación de PSC), publicada el 2002; Normativa de Factura Electrónica, publicada el 2003; y el Estándar de Documentos Electrónicos del Estado publicado el 2004 D.S 81.

La línea de trabajo "Certificados de firma electrónica" busca promover el uso masivo de certificados de clave pública en Chile, permitiendo transacciones electrónicas con igual respaldo que las realizadas en forma física sobre papel de acuerdo al marco jurídico nacional.

Las innovaciones más recientes se han enfocado en combinar técnicas de biometría con firma electrónica de clave pública, logrando un modelo de firma electrónica que no requiere que los titulares obtengan en forma previa un certificado de firma electrónica. Este modelo de materializo el año 2008 con la incorporación del nuevo servicio de Firma Electrónica Dactilar (FED), servicio que permite firmar acuerdos, contratos o convenios, o cualquier documento electrónico utilizando la huella dactilar como firma

electrónica simple. Para el año 2010, ya se firmaban mas de 60.000 documentos mensuales por esta vía, siendo uno de los servicios con mayor proyección en la empresa.

Como organización la empresa ha sufrido diversos cambios, cómo pasar de una planta de 8 personas en Febrero de 2000, a 85 a Octubre de 2011. Y pasar de 3 clientes a cerca de 12000 a la misma fecha.

El equipo de desarrollo comenzó como un equipo de 5 personas con dos proyectos a mediano plazo. Desarrollar la autoridad certificadora Acepta.com, y los servicios de factura electrónica.
La autoridad certificadora fue desarrollada usando un ciclo de vida en cascada, mientras para los servicios de factura electrónica se uso un desarrollo del tipo incremental.

Actualmente los servicios se encuentran funcionando, y se ha experimentado un incremento importante en la demanda por estos, lo que ha estado acompañado con un incremento en los reportes de Bugs.
A lo anterior se añade la intención de la empresa de generar nuevos productos e incursionar en nuevos mercados, lo que se ha traducido en la generación de nuevos proyectos de desarrollo de software con una alta prioridad.

Acepta.com ha sido uno de los actores más importantes en la definición y masificación de la factura electrónica en Chile. Esto la ha llevado a ser la marca más conocida y con más clientes dentro de este mercado. Con un 25% de participación en el mercado de empresas que eligen soluciones privadas de factura electrónica en Chile, supera por casi el doble a su competidor más cercano. (Datos basados en estudio de Penta Research durante diciembre del 2007)

La empresa también participa como líder en el mercado de emisión de certificados de firma electrónica en Chile, con un 39% de participación, deja al seguidor más cercano en un 27%. Esto se potencia por ser el único Prestador de Servicios de Certificación interconectado al Registro Civil. (Datos basados en un estudio hecho por Acepta.com al analizar los documentos tributarios electrónicos recibidos por sus clientes de factura electrónica)

La empresa organiza su gestión y operación a través de la siguiente estructura:

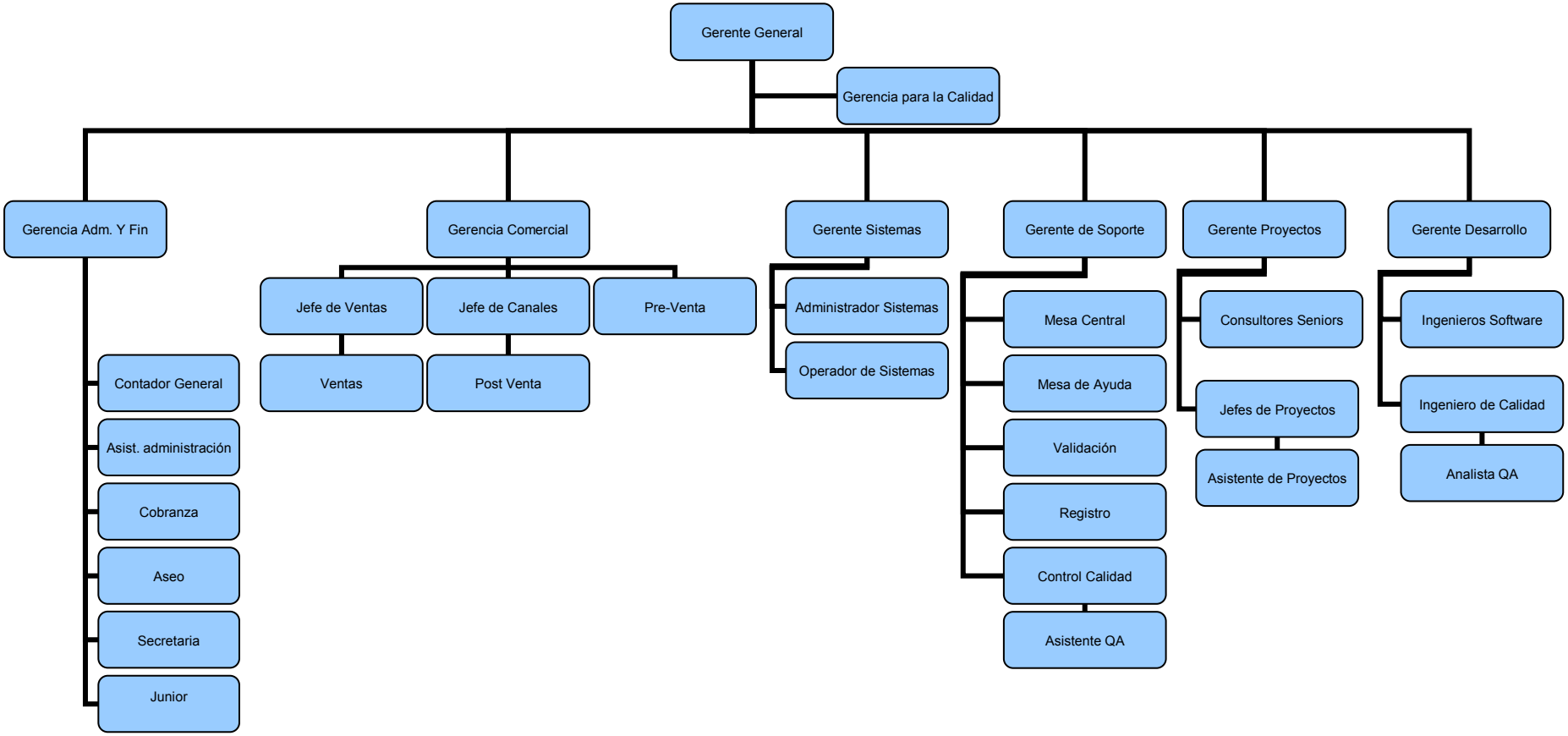


Figura 32: Organigrama Acepta.com

ANEXO B: COBERTURA DE PRUEBAS AUTOMÁTICAS INTRODUCIDAS

En los proyectos custodia-firma y ACM, se estructuro el software en mínimo tres módulos.

- **web**: implementa interfaz webui basado en jsf, y utiliza los servicios del módulo bean.
- **bean(o negocio)**: implementa las operaciones del negocio de la aplicación, y utiliza los servicios del módulo de persistencia, y otros que sean necesarios.
- **persistencia**: implementa la persistencia de datos.

La cobertura de pruebas para los proyectos custodia firma y ACM generada con maven a partir del plugin cobertura es la siguiente:

Custodia-firma.

Module	# Classes	Line coverage	Branch coverage	Complexity
custodia-firma-bean	17	69%	44%	1.626
custodia-firma-keystore	14	71%	70%	2.584
custodia-firma-persistencia	25	67%	55%	1.273
TOTAL	56	69%	60%	----------

Tabla 25: Resumen cobertura de pruebas custodia-firma

A continuación el desglose del informe generado con el plugin cobertura:

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	17	69%	288/414	44%	22/50	1.626
com.acepta.custodiafirma.bean	10	78%	140/179	40%	13/32	1.651
com.acepta.custodiafirma.pojo	1	100%	10/10	N/A	N/A	1
com.acepta.custodiafirma.servicios	6	61%	138/225	50%	9/18	1.69

Tabla 26: Cobertura de pruebas custodia-firma-bean

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	14	71%	303/424	70%	58/82	2.584
com.acepta.firma.xml	7	64%	155/239	69%	32/46	3.415
com.acepta.pki	7	80%	148/185	72%	26/36	1.875

Tabla 27: Cobertura de pruebas custodia-firma-keystore

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	25	67%	314/466	55%	11/20	1.273
com.acepta.custodiafirma.entities	12	73%	138/189	N/A	N/A	1
com.acepta.custodiafirma.persistencia	12	63%	155/243	58%	7/12	1.977
com.acepta.custodiafirma.persistencia.config	1	61%	21/34	50%	4/8	1.4

Tabla 28: Cobertura de pruebas custodia-firma-persistencia

ACM.

Module	# Classes	Line coverage	Branch coverage	Complexity
acepta-acm-bean	50	59%	46%	1.273
acepta-jobmanager-lib	6	62%	59%	1.676
acepta-acm-persistence	34	52%	15%	1.251
TOTAL	90	58%	45%	----------

Tabla 29: Resumen cobertura de pruebas ACM

A continuación el desglose del informe generado con el plugin cobertura:

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	50	59%	1052/1759	46%	174/378	1.842
com.acepta.acm.bean	1	0%	0/3	N/A	N/A	1
com.acepta.acm.bean.configuracion	3	69%	112/161	61%	38/62	2.259
com.acepta.acm.bean.importacion	4	83%	173/207	79%	35/44	2.31
com.acepta.acm.bean.importacion.parser	5	92%	58/63	N/A	N/A	1
com.acepta.acm.bean.importacion.parser.impl	4	88%	188/212	70%	14/20	1.72
com.acepta.acm.bean.job	8	69%	342/490	46%	63/136	2.519
com.acepta.acm.eventqueue	2	0%	0/89	0%	0/28	2.583
com.acepta.acm.generador	3	14%	2/14	50%	1/2	1.833
com.acepta.acm.generador.xsl	1	84%	11/13	N/A	N/A	2
com.acepta.acm.mail	7	56%	83/147	53%	14/26	2
com.acepta.acm.mail.listener	4	41%	19/46	16%	2/12	1.261
com.acepta.acm.mail.listener.impl	3	0%	0/80	N/A	N/A	1.286
com.acepta.acm.main	1	0%	0/84	0%	0/16	1.846
com.acepta.acm.pruebas	2	67%	33/49	50%	3/6	1.294
com.acepta.acm.reportes	1	79%	31/39	66%	4/6	1.3
com.acepta.acm.servicios	1	0%	0/62	0%	0/20	3.333

Tabla 30: Cobertura de pruebas acepta-acm-bean

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	6	62%	70/112	59%	25/42	1.676
com.acepta.jobsmanager	6	62%	70/112	59%	25/42	1.676

Tabla 31: Cobertura de pruebas acepta-jobmanager-lib

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	34	52%	430/822	15%	6/40	1.251
com.acepta.acm.persistence.entities	30	70%	331/470	N/A	N/A	1
com.acepta.acm.persistence.services	1	N/A	N/A	N/A	N/A	1
com.acepta.acm.persistence.services.impl	3	28%	99/352	15%	6/40	2.328

Tabla 32: Cobertura de pruebas acepta-acm-persistence

ANEXO C: EXPLORADOR DE PROCESOS (CATUTO)

Catuto es el nombre con el que se bautizo al explorador de procesos desarrollado como parte de esta tesis da respuesta a los requerimientos funcionales RF01, hasta el RF-10 definidos en 4.1.3.1.1 y los requerimientos no funcionales definidos en 4.1.3.1.2.

Para satisfacer los requerimientos de seguridad expresados en 4.1.3.1.2 referidos a la autenticación sobre un directorio LDAP, se implemento una extensión al motor de procesos Activiti de modo de reemplazar el modulo de seguridad que trae incorporado y utilizar uno propio.

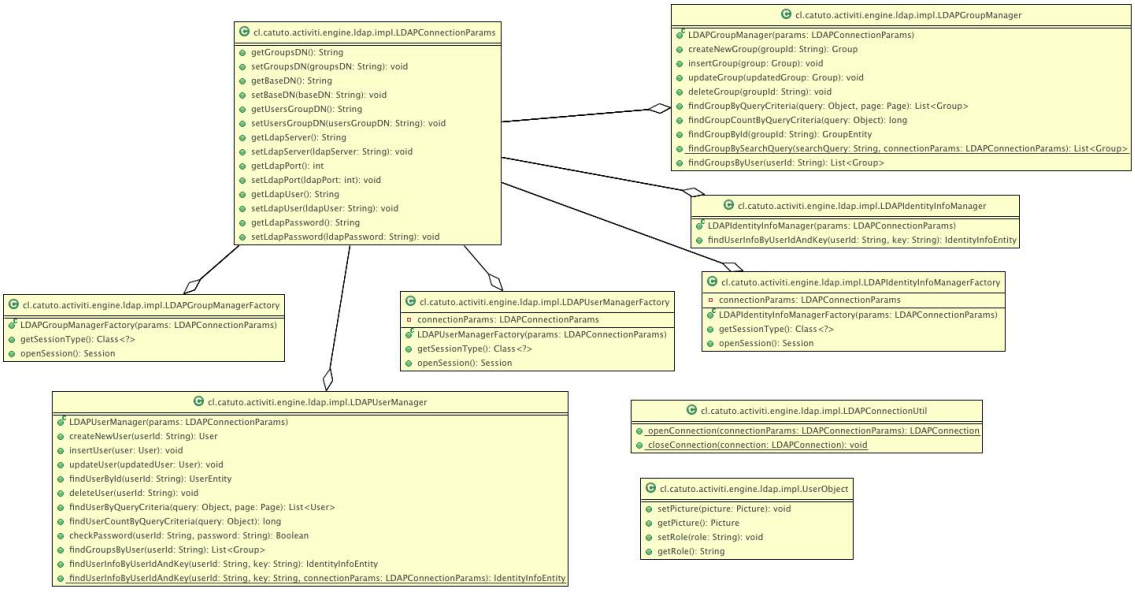


Figura 33: Diagrama de clases extensión LDAP para Activiti

Esta extensión se habilita a través de una simple configuración en el archivo activiti.cfg.xml, el cual una vez configurado permite utilizar la api de Activiti e manera transparente.

```
<bean id="processEngineConfiguration" class="org.activiti.cdi.CdiJtaProcessEngineConfiguration">
    <property name="history" value="full" />
    <property name="dataSourceJndiName" value="jdbc/activiti" />
    <property name="databaseType" value="postgres" />
    <property name="transactionManager" ref="transactionManager" />
    <property name="transactionsExternallyManaged" value="true" />
    <property name="databaseSchemaUpdate" value="true" />

    <property name="mailServerHost" value="smtp.gmail.com" />
    <property name="mailServerPort" value="587" />
    <property name="mailServerUseTLS" value="true" />
    <property name="mailServerDefaultFrom" value="catuto@millalen.cl" />
    <property name="mailServerUsername" value="catuto@millalen.cl" />
    <property name="mailServerPassword" value="catuto2012" />

    <property name="customSessionFactories">
      <list>
        <bean class="cl.catuto.activiti.engine.ldap.impl.LDAPUserManagerFactory">
          <constructor-arg ref="ldapConnectionParams" />
        </bean>
        <bean class="cl.catuto.activiti.engine.ldap.impl.LDAPGroupManagerFactory">
          <constructor-arg ref="ldapConnectionParams" />
        </bean>
        <bean class="cl.catuto.activiti.engine.ldap.impl.LDAPIdentityInfoManagerFactory">
          <constructor-arg ref="ldapConnectionParams" />
        </bean>
      </list>
    </property>

</bean>
```

Figura 34: Configuración LDAP como repositorio de usuarios y permisos

Una vez configurado el servidor LDAP, el usuario se puede acceder al sistema identificándose con su usuario y password corporativa

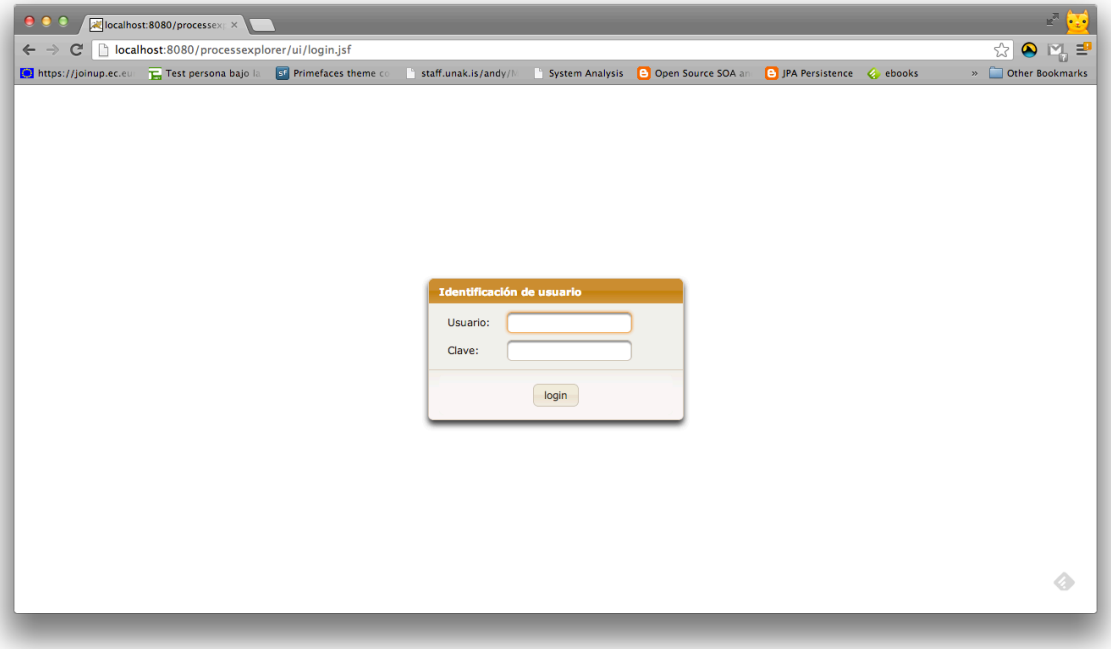


Figura 35: Pantalla de login Catuto Explorer

A continuación las funciones del sistema de gestión de procesos accesibles por el usuario una vez autenticado.

- Iniciar un nuevo caso de alguno de los procesos desplegados en el sistema.

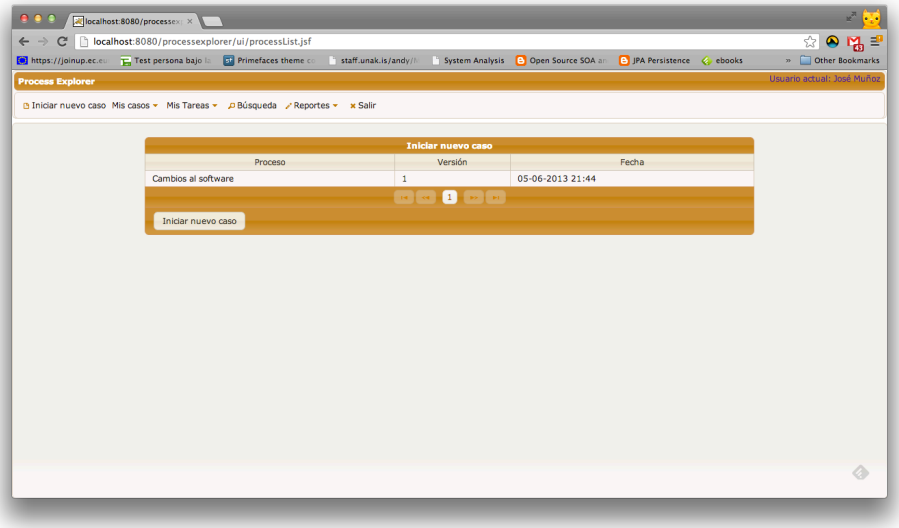


Figura 36: Pantalla para inicio de una nueva instancia de proceso

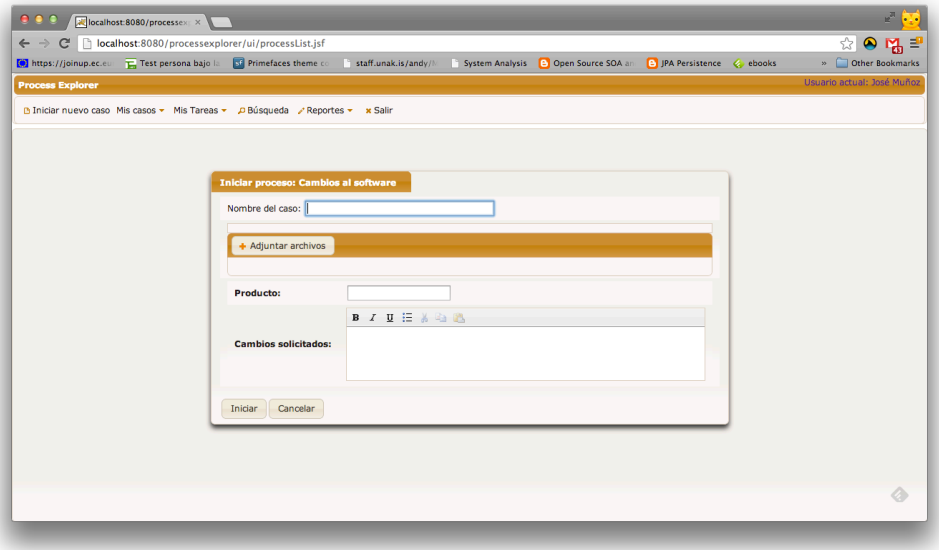


Figura 37: Formulario de inicio de una nueva instancia de proceso

- Consulta de procesos iniciados y/o supervisados por el usuario

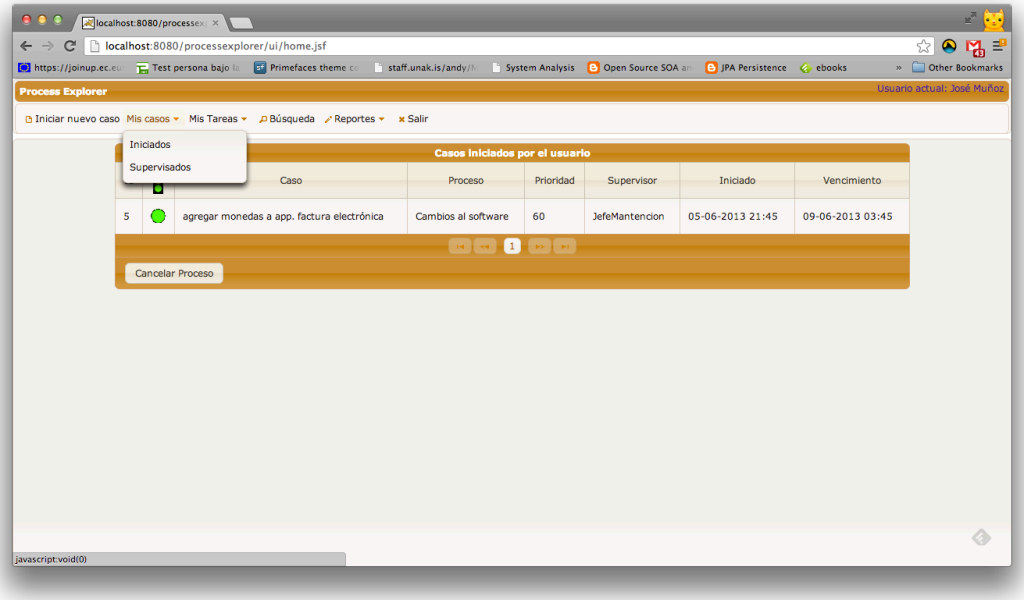


Figura 38: Consulta sobre los casos iniciados o supervisados del usuario

- Consulta de casos pendientes, completados o cancelados del usuario.

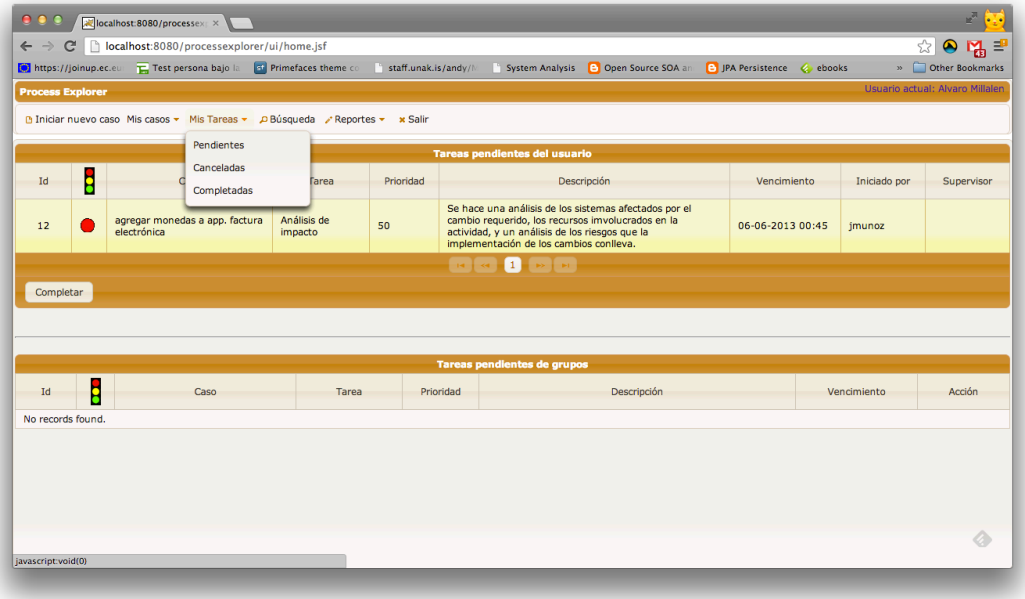


Figura 39: Consulta sobre los casos pendientes, completados o cancelados por el usuario

- Consulta y búsqueda de casos utilizando texto libre

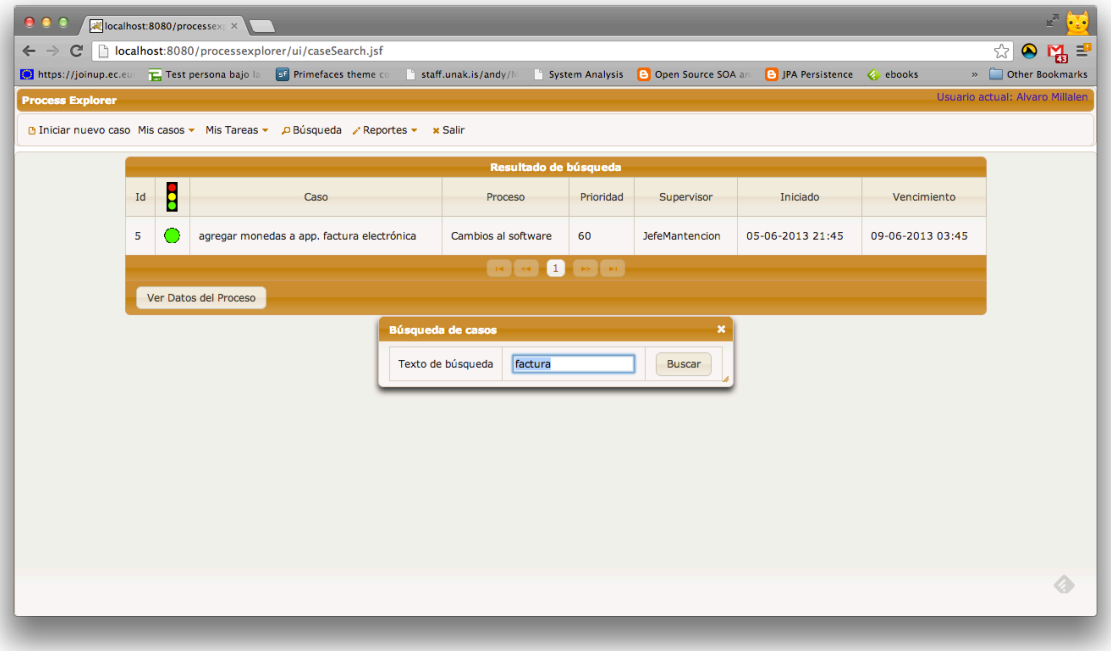


Figura 40: Búsqueda de casos por texto libre

- Monitoreo del desempeño de los procesos

Para permitir el monitoreo de los procesos y su evaluación , se definieron KPI estándares para los procesos.

Carga del proceso: Este indicador la cantidad de instancias del proceso en las categorías

- A tiempo: Aquellas instancias en ejecución que se encuentran en los tiempos esperados.
- Atrasado: Aquellas instancias en ejecución que se encuentran más allá de los tiempos esperados.
- En peligro: Aquellas instancias en ejecución que se encuentran

Casos en progreso: Este indicador muestra los porcentajes de casos en las categorías definidas anteriormente.

Casos por expirar: Este indicador muestra los casos por expirar en los próximos tres días.

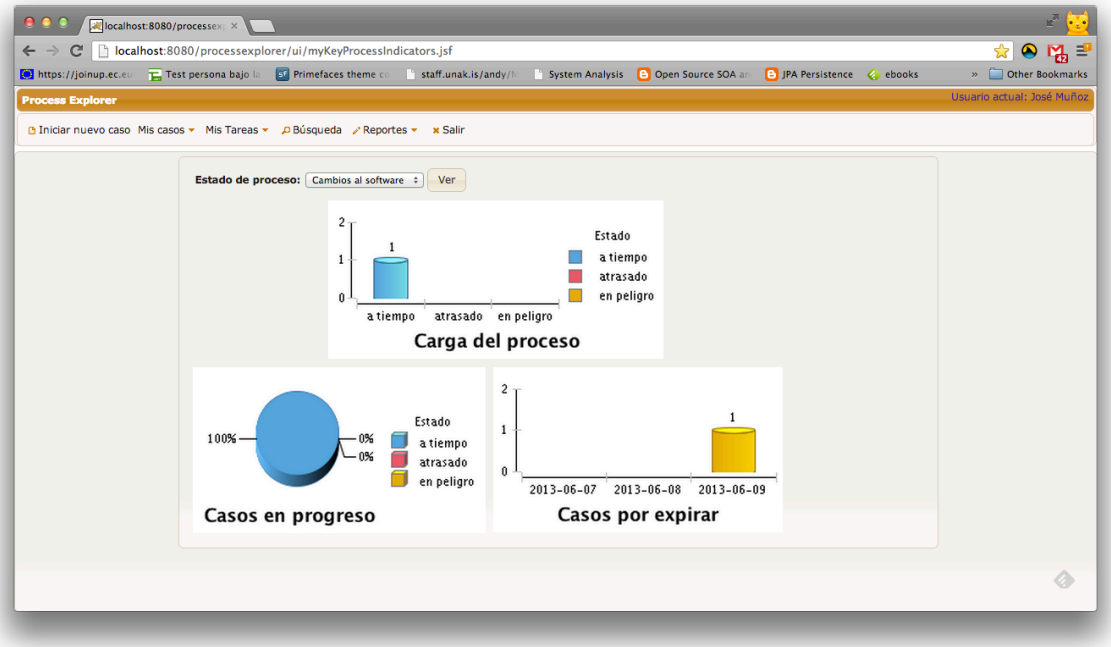


Figura 41: Consulta de KPI de procesos

ANEXO D: HERRAMIENTA DE DISEÑO BPMN Y EXTENSIONES

Para dar soporte a estos indicadores se necesita establecer SLA para los procesos y para las actividades, para lo cual se opto por hacer una extensión a bpmn y agregar un atributo para establecer el SLA esperado para un proceso y sus actividades, y es parte del diseño del proceso.
Así un proceso bpmn con sla establecido luciria como :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:activiti="http://a
  <process catuto:managedBy="JefeMantencion" catuto:sla="(3,6,0)" id="CambiosDeSoftware"
    <startEvent id="_2" isInterrupting="true" name="Start Event" parallelMultiple="false">
```

Figura 42: Extensiones a BPMN

Donde las extensiones propietarias añadidas tiene el prefijo catuto.

catuto:managedBy: Establece el rol que tendra a cargo la supervisión de este proceso.
catuto:sla: Establece el tiempo esperado de duración del proceso. El valor es de la forma (d,h,m), don d es dias, h horas y m, minutos.

Para facilitar su incorporación en el proceso de diseño de procesos, se implementó un plugin para extener la funcionalidad Yaoqiang.

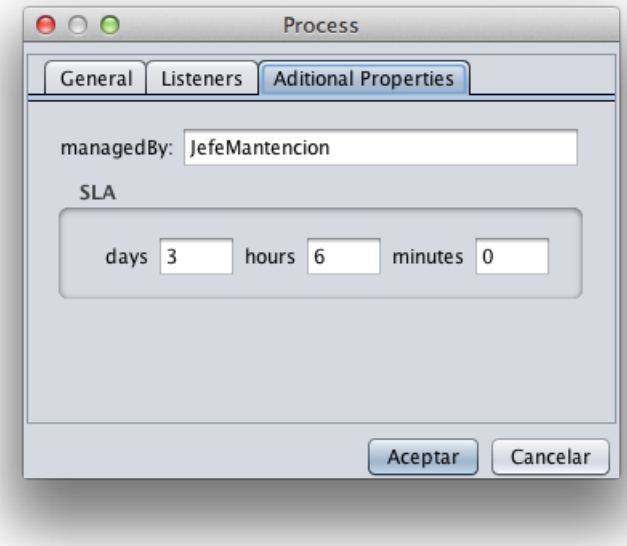


Figura 43: Extensiones para procesos en la herramienta de diseño

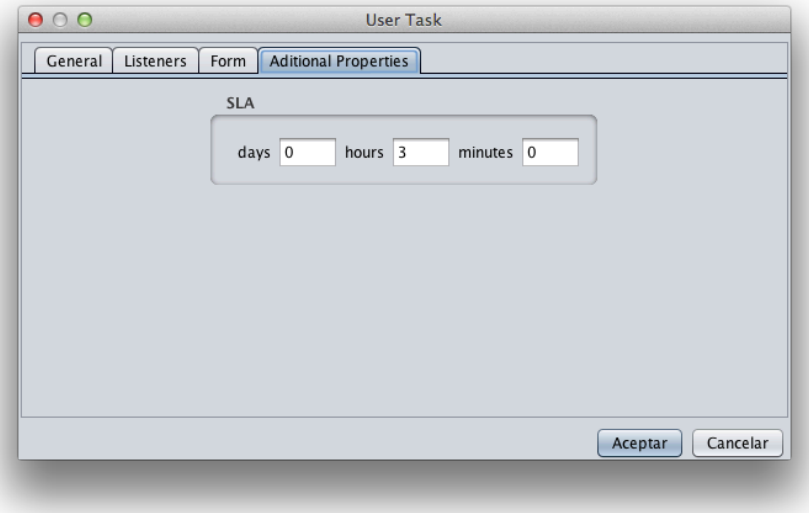


Figura 44: Extensiones para tareas en herramienta de diseño

ANEXO E: HERRAMIENTAS DE APOYO UTILIZADAS EN LA AUTOMATIZACIÓN DE FLUJOS

Para llevar a cabo la solución se necesitaron herramientas de apoyo para el diseño de los procesos. A continuación se presentan las herramientas utilizadas, las cuales se enmarcan en la propuesta de utilización de tecnologías y técnicas de BPM.

Eclipse v.3.7.2: IDE para el desarrollo de aplicaciones en JEE, Python, C, php, entre otros. Este producto se utilizó para el desarrollo de una aplicación que de soporte a la automatización de procesos.

Activiti Eclipse BPMN 2.0 designer: Plug-In para eclipse para agregar al IDE la posibilidad de crear y editar procesos en BPMN 2.0. Este producto fue utilizado en las primeras etapas del desarrollo de la solución.

Yaoqiang BPMN Editor v.2.1.24: Editor BPMN 2.0 open source liviano, fácilmente extensible para soportar características adicionales sobre BPMN. Esta herramienta se utilizó para dar soporte a extensiones implementadas para la automatización de los procesos propuestos.

BonitaStudio v.5.10: Editor y simulador BPMN 2.0. Esta herramienta se utilizó para realizar la simulación de los modelos de los procesos actuales y propuestos, de modo de poder comparar su rendimiento en situaciones de carga similares.

BIzAgi Process Modeler v2.5.1.1: Modelador y simulador de procesos BPMN 2.0. Esta herramienta se utilizó para simulación de los modelos de los procesos actuales y propuestos, de modo de poder compara su rendimiento en situaciones de carga similares.

En este contexto, la herramienta que resulto más relevante fue el simulador de procesos integrado con BizAgi, pues este permitió hacer una validación de los procesos propuestos, dada la imposibilidad de probar todos los cambios en el ambiente de producción.

ANEXO F: MOTOR DE PROCESOS ACTIVITI

Activiti es un motor de procesos ligero para java, que permite la ejecución de procesos definidos mediante la especificación BPMN 2.0.

El motor de procesos Activiti es esencialmente una máquina de estados. Cuando un proceso es desplegado sobre el motor de procesos, y se inicia una nueva instancia de un proceso, los elementos BPMN 2.0 son ejecutados uno por uno. Esto es muy similar a una máquina de estados, donde existe un estado activo, y basado en condiciones el estado en ejecución cambia a otro estado a través de transiciones.

La forma de interactuar con Activiti es a través de sus APIs, el punto central de inicio es ProcessEngine, desde el cual se pueden obtener los diferentes servicios para la ejecución, monitoreo, y autenticación.

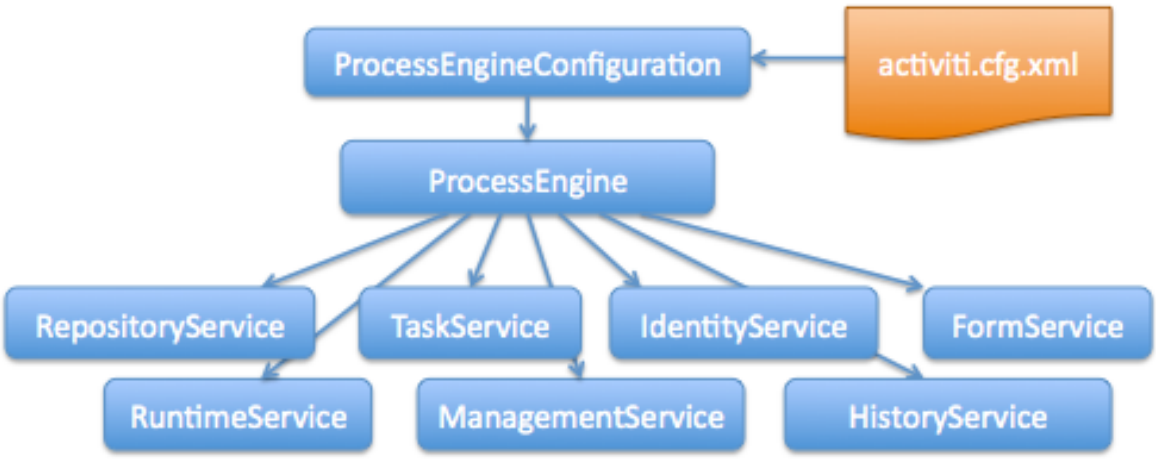


Figura 45: Componentes de la API de Activiti

activiti.cfg.xml: archivo de configuración de Activiti donde se especifica principalmente la persistencia de los datos del motor.

ProcessEngineConfiguration: Representa la configuración actual del motor de procesos.

ProcessEngine: Es el motor de procesos propiamente tal, el que expone siete servicios para la administración de los procesos y tareas asociadas a el.

RepositoryService: Este servicio provee la funcionalidad para desplegar, consultar borrar, y recuperar definiciones de procesos.

TaskService: Este servicio provee la funcionalidad para preguntar por las tareas asignadas a usuarios o a grupos, manipular a que usuario es asignada una tarea, o que usuarios están involucrados de alguna manera en una tarea.

IdentityService: Este servicio provee la funcionalidad para la administración de usuarios y grupos. Este servicio trae una implementación local, la que puede ser reemplazada para que la base de este servicio sea un servidor LDAP, u otro medio para la autenticación y administración de permisos de usuario.

FormService: Este es un servicio opcional, lo que significa que Activiti puede ser usado sin el. Este servicio introduce los conceptos de formulario de inicio y formulario de tarea. Activiti permite la definición de estos formularios en BPMN 2.0 y a través de este servicio permite acceder a estos datos.

RuntimeService: Este servicio provee una interfaz para iniciar, y preguntar por instancias de proceso. Además, se puede recuperar los valores de variables de proceso, y cambiar sus valores.

ManagementService: Este servicio puede ser utilizado para hacer consultas sobre las tablas de Activiti y ejecutar tareas de administración.

HistoryService: Este servicio permite recuperar información referente a instancias de proceso ya completadas.

ANEXO G: SELECCIÓN HERRAMIENTA DE SIMULACIÓN.

La selección de la herramienta de simulación se realizo mediante la evaluación de herramientas del tipo opensource y freeware. Las requerimientos básicos buscados en la herramienta son:

- Soporte de estándar BPMN 2.0
- Estabilidad de ejecución del programa.
- Manejo de recursos asociados a la ejecución de las tareas definidas.
- Manejo de calendario de disponibilidad de los recursos asociados a las tareas definidas.
- Generación de reportes de resultados.
- Análisis de caminos alternativos.
- Soporte de para distribuciones estadísticas que caractericen las entradas al sistema.

La selección se realizo en dos iteraciones. En la primera iteración se evaluó jBPM, BonitaStudio y BIMP Simulator. El resultado comparativo de las pruebas realizadas en cada uno de los software se aprecia en la siguiente tabla:

	BPMN2.0	Estabilidad de ejecución	Manejo de recursos	Calendario de recursos	Reportes	Análisis de Caminos	Distribución Estadística
jBPM v5.4.0	✓	✗	✗	✗	✓	✓	✓
Bonita v5.10	✓	✓	✓	✓	✓	✗	✗
BIMP Simulator	✓	✗	✓	✓	✓	✗	✓

Tabla 33: Resultado iteración 1, selección de simulador bpmn

jBPM, resulto poco estable durante su ejecución, sufriendo de frecuentes errores de aplicación. No maneja calendario para los recursos, y los recursos solo se expresan como un numero a nivel de tareas, sin la posibilidad de compartir recursos entre tareas, y los reportes a nivel de tareas no entregan información detallada. A su favor tiene soporte para BPMN2, reportes a nivel de tareas, a nivel de caminos posibles de ejecución. Posee soporte para las distribuciones normal, uniforme, y de poisson a nivel de tareas. Para la entrada de eventos al sistema solo se define el intervalo entre instancias de eventos, no pudiéndose comprobar si en este caso se usa una distribución exponencial o se distribuyen constantes en el tiempo de simulación.

BIMP Simulator, resulto poco estable durante su ejecución, sufriendo de frecuentes errores de aplicación. A su favor maneja definición de recursos se definen de manera global al proceso, incluyendo calendario, para luego ser asignados a las tareas asociadas. Tiene soporte para BPMN2, reportes a nivel de tareas, y de proceso. Posee soporte para las distribuciones normal, exponencial, y uniforme.

Bonita, resulto estable durante su ejecución. Maneja definición de recursos se de manera global al proceso, incluyendo calendario, los que luego pueden ser asignados a las tareas asociadas. Tiene soporte para BPMN2, reportes a nivel de tareas, y de proceso. No posee soporte para distribuciones estadísticas, en su lugar, las entradas de eventos se definen a través de perfiles de inyección de eventos, pudiendo inyectarse los eventos todos a la vez o distribuidos de manera constante en el periodo de tiempo definido.

En estas condiciones, se opto por utilizar BonitaStudio.

Durante el desarrollo de este trabajo se realiza una nueva evaluación de la herramienta, considerando la liberación de un software BPMN freeware con capacidades de simulación BizAgi v2.5.1.1.

La tabla comparativa queda en esta segunda iteración, de la siguiente manera:

	BPMN2.0	Estabilidad de ejecución	Manejo de recursos	Calendario de recursos	Reportes	Análisis de Caminos	Distribución Estadística
jBPM v5.4.0	✓	✗	✗	✗	✓	✓	✓
Bonita v5.10	✓	✓	✓	✓	✓	✗	✗
BIMP Simulator	✓	✗	✓	✓	✓	✗	✓
BizAgi v.2.5.1.1	✓	✓	✓	✓	✓	✗	✓

Tabla 34: Resultado iteración2, selección de simulador bpmn

BizAgi, es un modelador freeware que en la versión 2.5.5.1, incorpora un motor de simulación a través de una alianza con la empresa Lanner, nacida en 1996, y especialista en software para simulación de procesos de negocio.
El simulador se integra de modo transparente con el modelador, no encontrándose problemas de estabilidad del software mientras se realizaron las pruebas. El software provee manejo de recursos de manera global al proceso, con capacidad de definir calendarios y asociarlos a los recursos al momento de establecer los recursos de las tareas del proceso. En software entrega un reporte simple, con información suficiente para evaluar la performance del proceso. Las distribuciones estadísticas soportadas son: Normal, Truncada Normal, Triangular, Uniforme, LogNormal, Beta, Exponencial, Gamma, Erlang, Weibull, Binomial, Poisson. Además soportar las especificación de distribuciones constantes.

Dado este escenario, se decidió cambiar la herramienta de simulación a Bizagi, por la confiabilidad que da el software que soporta la funcionalidad, y el soporte a distribuciones estadísticas que se acomodan de mejor manera a la naturaleza del proceso que se busca simular.

Cabe mencionar que la diferencia observada entre los resultados entregados por Bonita y los entregados por BizAgi es importante en cuanto a la magnitud de los tiempos de proceso, por lo que un factor importante a considerar en el futuro es la madurez de la solución de simulación, si es que esta se pretende incorporar como una práctica habitual en el modelamiento y mejora de procesos.

ANEXO H: CASOS DE USO PARA HERRAMIENTA GESTION DE PROCESOS.

Para cada uno de los requerimientos funcionales identificados se especifican sus correspondientes casos de uso:

Identificador:	CU01
Nombre:	Desplegar nuevo proceso
Actores:	Administrador
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	El proceso queda registrado en el sistema
Flujo normal de los eventos:	
1. El actor selecciona la opción Desplegar nuevo proceso 2. El sistema muestra pantalla con formulario de despliegue 3. El actor selecciona el archivo de proceso 4. El actor confirma los datos ingresados 5. El sistema registra nuevo proceso 6. El sistema confirma ingreso de nuevo proceso	
Flujo alternativos:	
El actor cancela la operación En el paso 3, o paso 4 el actor cancela la operación, se cancela la transacción del sistema vuelve a su estado inicial.	
Excepciones:	
Faltan datos En el paso 4 faltan campos obligatorios para completar el registro. Se muestra el mensaje de advertencia y se regresa a la pantalla indicando los campos que faltan por completar.	

Tabla 35: Caso de uso desplegar nuevo proceso

Identificador:	CU02
Nombre:	Iniciar caso
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	El caso queda registrado en el sistema
Flujo normal de los eventos:	
1. El actor selecciona la opción Iniciar nuevo caso 2. El sistema muestra pantalla con formulario inicial 3. El actor llena los campos requeridos 4. El actor confirma los datos ingresados 5. El sistema registra nuevo caso 6. El sistema confirma ingreso de nuevo caso	
Flujo alternativos:	
El actor cancela la operación En el paso 3, o paso 4 el actor cancela la operación, se cancela la transacción del sistema vuelve a su estado inicial.	
Excepciones:	
Faltan datos En el paso 4 faltan campos obligatorios para completar el registro. Se muestra el mensaje de advertencia y se regresa a la pantalla indicando los campos que faltan por completar.	

Tabla 36: Caso de uso iniciar caso

Identificador:	CU03
Nombre:	Completar tarea pendiente
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	Tarea seleccionada se registra como completada.
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis tareas pendientes 2. El sistema muestra pantalla la lista de tareas pendientes del actor 3. El actor selecciona tarea a completar 4. El sistema muestra formulario con los datos a ser completados por el actor 5. El actor llena los campos requeridos 6. El actor confirma acción de “Completar tarea” 7. El sistema registra tarea completada 8. El sistema confirma tarea como completada	
Flujo alternativos:	
El actor cancela la operación En el paso 4, o paso 5 el actor cancela la operación, se cancela la transacción del sistema y vuelve a su estado inicial.	
Excepciones:	
Faltan datos En el paso 6 faltan campos obligatorios para completar el registro. Se muestra el mensaje de advertencia y se regresa a la pantalla indicando los campos que faltan por completar.	

Tabla 37: Caso de uso completar tarea pendiente

Identificador:	CU04
Nombre:	Revisar casos iniciados
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	Se despliega lista con estado de casos iniciados por el actor
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis casos iniciados 2. El sistema muestra pantalla la lista de casos iniciados por el actor	
Flujo alternativos:	
No aplica.	
Excepciones:	
Faltan datos No aplica.	

Tabla 38: Caso de uso revisar casos iniciados

Identificador:	CU05
Nombre:	Revisar casos supervisados
Actores:	Supervisor
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	Se despliega lista con estado de casos supervisados por el actor
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis casos supervisados 2. El sistema muestra pantalla la lista de casos supervisados por el actor	
Flujo alternativos:	
No aplica.	
Excepciones:	
Faltan datos No aplica.	

Tabla 39: Caso de uso revisar casos supervisados

Identificador:	CU06
Nombre:	Revisar tareas completadas
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	Se despliega lista de tareas completadas por el actor.
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis tareas completadas 2. El sistema muestra pantalla la lista de tareas completadas por el actor	
Flujo alternativos:	
No aplica.	
Excepciones:	
Faltan datos No aplica.	

Tabla 40: Caso de uso revisar tareas completadas

Identificador:	CU07
Nombre:	Revisar tareas canceladas
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	Se despliega lista de tareas canceladas asignadas al actor.
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis tareas canceladas 2. El sistema muestra pantalla la lista de tareas canceladas asignadas al actor	
Flujo alternativos:	
No aplica.	
Excepciones:	
Faltan datos No aplica.	

Tabla 41: Caso de uso revisar tareas canceladas

Identificador:	CU08
Nombre:	Cancelar caso iniciado
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	El caso queda registrado como cancelado
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis casos iniciados 2. El sistema muestra pantalla la lista de casos iniciados por el actor 3. El actor selecciona caso 4. El sistema muestra formulario de cancelación de caso 5. El actor confirma la acción de “Cancelar caso” 6. El sistema registra caso como cancelado 7. El sistema confirma caso como cancelado	
Flujo alternativos:	
El actor cancela operación. En el paso 5, el actor cancela la operación, el sistema cancela la transacción, y el sistema vuelve a su estado inicial.	
Excepciones:	
Faltan datos En el paso 5, faltan campos obligatorios para completar la operación. Se muestra el mensaje de advertencia y se regresa a la pantalla indicando los campos que faltan por completar.	

Tabla 42: Caso de uso cancelar caso iniciado

Identificador:	CU09
Nombre:	Cancelar caso supervisado
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	El caso queda registrado como cancelado
Flujo normal de los eventos:	
1. El actor selecciona la opción Mis casos supervisados 2. El sistema muestra pantalla la lista de casos supervisados por el actor 3. El actor selecciona caso 4. El sistema muestra formulario de cancelación de caso 5. El actor confirma la acción de “Cancelar caso” 6. El sistema registra caso como cancelado 7. El sistema confirma caso como cancelado	
Flujo alternativos:	
El actor cancela operación. En el paso 5, el actor cancela la operación, el sistema cancela la transacción, y el sistema vuelve a su estado inicial.	
Excepciones:	
Faltan datos En el paso 5, faltan campos obligatorios para completar la operación. Se muestra el mensaje de advertencia y se regresa a la pantalla indicando los campos que faltan por completar.	

Tabla 43: Caso de uso cancelar caso supervisado

Identificador:	CU10
Nombre:	Ver estado de procesos
Actores:	Usuario
Precondiciones:	El actor se ha identificado exitosamente en el sistema
Poscondiciones:	El sistema muestra indicadores de procesos
Flujo normal de los eventos:	
1. El actor selecciona la opción Reportes/Estado de procesos 2. El actor selecciona proceso a revisar 3. El sistema muestra los indicadores de estado del proceso	
Flujo alternativos:	
No aplica	
Excepciones:	
No aplica	

Tabla 44: Caso de uso ver estado de procesos

Identificador:	CU11
Nombre:	Diseñar nuevo proceso
Actores:	Supervisor
Precondiciones:	El actor tiene instalada aplicación de diseño de procesos.
Poscondiciones:	La aplicación genera archivo de definición de proceso.
Flujo normal de los eventos:	
1. El actor inicia aplicación 2. El actor diseña los flujos del proceso 3. El actor salva archivo de proceso.	
Flujo alternativos:	
El actor modifica proceso preexistente. En el paso 2, el actor carga definición de proceso anterior, y realiza las modificaciones de flujo necesarias.	
Excepciones:	
No aplica	

Tabla 45: Caso de uso diseñar nuevo proceso

ANEXO I: MANUAL DE MANTENCIÓN Y OPERACIÓN.

Habitualmente los productos tienen documentación para el usuario final, sin cubrir a quienes deben operar las soluciones o quienes deben intervenirlas en el futuro. Este documento viene a cubrir este vacío.

Este documento se ha incorporado como parte de la etapa de transferencia desde el área de desarrollo a los responsables de la mantención y/u operación de los sistemas.

El contenido de este documento se basa en una plantilla original del cdcup [19], y es básicamente:

1 INTRODUCCIÓN
1.1 Propósito
1.2 Audiencia
2 DESCRIPCIÓN DEL SISTEMA
2.1 Características clave/principales
2.2 Inventario
2.3 Ambiente
2.4 Operación del sistema
2.5 Arquitectura del sistema
3 INSTALACIÓN DE LA APLICACIÓN
3.1 Usuarios nuevos
3.2 Controles de acceso
3.3 Instalación
3.4 Configuración
3.5 Inicio del sistema
3.6 Parada del sistema
3.7 Suspensión del sistema
4 USO DEL SISTEMA
4.1 Instrucciones
4.2 Convenciones y mensajes de error
5 ADMINISTRACIÓN DEL SISTEMA
5.1 Administración del cambio
5.2 Administración de la configuración
5.3 Administración de release
5.4 Administración de la seguridad
5.5 Administración del sistema
6 ADMINISTRACIÓN DEL SERVICIO
7 REQUERIMIENTOS REGULATORIOS
8 FAQS

Esta es sólo una plantilla, y el contenido se llena según el caso, e incluso puede ser una wiki. Si el contenido ya existe en un documento anterior solo se agrega una referencia.

A través de este contenido se proporciona la información mínima de operación de los sistemas, y se deja explicito el modo en que se gestiona tanto el sistema como los servicios que este otorga. Además se explicita las diferentes facetas de la administración del sistema una vez en operación.