Universidad de las Ciencias Informáticas

Título: Guía metodológica para gestionar la integración de componentes en los proyectos del sistema CEDRUX

Trabajo de Diploma para optar por el grado de Máster en Ciencias Técnicas

Autor: Nemury Silega Martínez

Tutor: Msc. Yadenis Piñero Pérez

Septiembre 2010

Agradecimientos

Especiales agradecimientos a mi tutora Yadenis, que ha sido la guía permanente para lograr esta tesis y es la principal artífice de este trabajo.

Les agradezco a todos mis compañeros del proyecto ERP, en especial a los de la línea de Finanzas y a los que me ayudaron directamente cuando lo necesité.

A mi amigo Carlos por ayudarme tanto con la búsqueda de libros.

Les agradezco a todas las personas que me han ayudado durante este tiempo de trabajo en la tesis.

Dedicatoria

Se la dedico a:

En primer lugar a lo más grande que tengo en mi corazón, que es mi madre.

La luz de todos mis días, mi novia buena y hermosa Dana.

Mi tía del alma, Julia, mi fuente de inspiración permanente.

Mi hermano del corazón y a mis dos sobrinitos.

Mi papá y mi hermana, que tanto los quiero.

Mi amigo, padre y hermano Gustavo

Mis tías Gina, Hilda, Juanita.

Mis primas Jenny y Aliuska.

Mi primo Yorlenys .

Mis suegros Juan y María que tanto me apoyan y ayudan.

Resumen

Uno de los problemas comunes en el desarrollo de software por componentes es el relacionado con la integración, los componentes se desarrollan de forma independiente y luego tienen que ser integrados. En el desarrollo del sistema CEDRUX se presentan diversos problemas causados por la deficiente gestión de la integración de componentes, influidos además por la estructura organizativa que tiene el proyecto. En la presente investigación se presenta una propuesta para gestionar el proceso de integración de componentes en el sistema CEDRUX.

INDICE

Introducción

Los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés), son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Los ERP son sistemas integrales de gestión para la empresa. Se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación. Estas partes son de diferente uso, por ejemplo: producción, ventas, compras, logística, contabilidad (de varios tipos), gestión de proyectos, GIS (sistema de información geográfica), inventarios y control de almacenes, pedidos, nóminas, entre otros. Sólo se puede definir un ERP como la integración de todas las parte(Carlos Menezes 2003).

Determinado por las condiciones de Cuba y considerando los grandes beneficios de un sistema de este tipo se decidió desarrollar un ERP, al cual se le ha denominado CEDRUX y debe satisfacer las condiciones de Cuba. El sistema se desarrolla en la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Informatización de la Gestión de Entidades (CEIGE). Según la definición expuesta, este tipo de sistema gestiona un número importante de procesos empresariales, lo que lo convierte en un sistema de gran magnitud donde la gestión de los procesos son traducidos en subsistemas y componentes en el sistema. Generalmente el desarrollo de un ERP lleva mucho tiempo y se especializan en algunas ramas, por ejemplo ERP para hospitales, ERP para fábricas, entre otros. Sin embargo el ERP CEDRUX tiene que ser desarrollado en un corto período de tiempo si se compara con su magnitud. Además debe ser lo más general posible y adaptarse a todas las entidades de Cuba, independientemente de su objeto social. Pese a estos grandes requerimientos para el desarrollo de CEDRUX se cuenta con personal que no siempre tiene la experiencia en el desarrollo de sistemas complejos. Entre los integrantes del proyecto se encuentra un gran número de estudiantes y de profesionales que desarrollan otras actividades además de la producción entre las que se encuentran la docencia, investigación y extensión universitaria.

El proyecto está dividido en líneas de desarrollo. En cada línea existen varios proyectos, donde se desarrollan varios subsistemas los cuales están constituidos por componentes. Cada proyecto tiene cierto nivel de independencia en la gestión de proyecto aunque debe cumplir con lineamientos que se define a nivel superior. Pero generalmente la gestión en una línea puede que no coincida con otras. Algo que complica un poco la gestión es que los componentes

de un proyecto están conectados con otros que pueden no pertenecer al mismo proyecto. En la Figura 1 se puede apreciar la organización descrita.

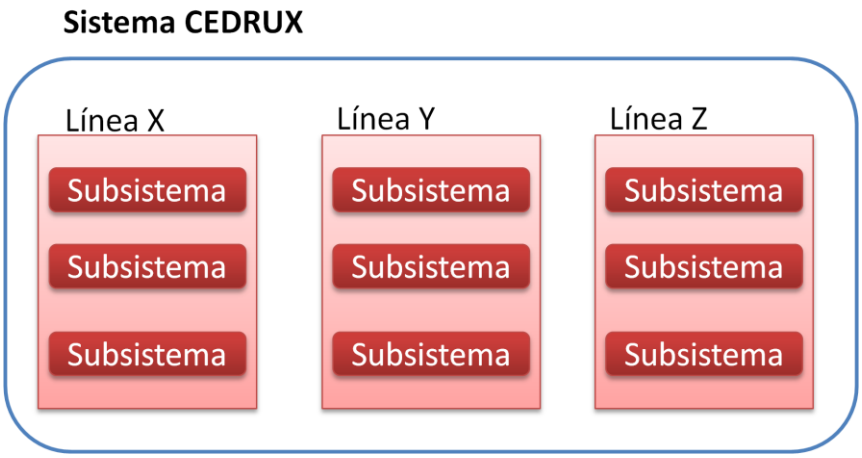


Figura 1 Organización del desarrollo del sistema CEDRUX

El CEDRUX se desarrolla por componentes, aprovechando las ventajas que posee este tipo de desarrollo. Dentro de las principales ventajas se encuentran la posibilidad de reutilizar componentes que han sido probados y utilizados en otros contextos, además de permitir el desarrollo de un sistema mediante el ensamblado de varios componentes y que puedan ser desarrollados por equipos diferentes y de forma independiente, entre otras.

Con este tipo de desarrollo, pese a existir grandes ventajas se presentan algunas situaciones que influyen negativamente en la gestión de los proyectos de CEDRUX. Los componentes son desarrollados de forma independiente y deben integrarse para que el sistema funcione, pero cuando un componente necesita consumir algún servicio de otro, el proceso puede volverse engorroso. Uno de los principales motivos está determinado por el hecho de que no existe por proyecto una persona encargada de gestionar los servicios que brindan y consumen los componentes del proyecto. Determinando que la comunicación sea muy complicada entre varios proyectos, incluso dentro de un mismo proyecto, además provoca que cuando sean necesarias nuevas integraciones o cuando existan irregularidades en el proceso de integración no existe una persona responsable del tema.

Otro elemento negativo en el proceso es que no está formalmente definido el flujo de actividades que deben desarrollarse cuando un componente necesita integrarse. Por lo que cuando se identifican necesidades de

integración se carece de mecanismos formales para la solicitud de servicios en caso que sea entre proyectos diferentes, tampoco existen mecanismos para gestionar la integración interna de un proyecto. Esta situación provoca que en cada proyecto estas actividades se desarrollen intuitivamente y diferente en cada proyecto lo que generalmente provoca desviaciones en el desarrollo. Además como no se conocen con anterioridad las tareas relacionadas con la integración no pueden ser planificadas, lo que convierte la planificación del desarrollo de un componente en irreal y generalmente por causa de integración se generan desviaciones significativas, porque se ve el tema de integración como un trabajo extra al desarrollo del componente.

Los artefactos que existen en este momento vinculados con la integración no brindan la información suficiente para que el proceso se desarrolle satisfactoriamente y que por cada componente se tengan los datos de integración. Este elemento dificulta el proceso de integración pues es muy complicado saber todos los servicios que brinda y consume un componente, también influye en las decisiones de reutilización y portabilidad de los componentes y provoca que cuando existen cambios en un componente y en sus servicios sea muy complicado determinar el impacto en el sistema. Además como no se tiene un registro de dependencia, cuando un componente no funciona correctamente; es habitual que no se conozca en principio si el problema es propio de él o de algún servicio de los que consume.

No se tienen mecanismos de retroalimentación eficientes que permitan identificar rápidamente los errores introducidos en los componentes que afecten la integración. Lo que provoca que los errores sean detectados mucho tiempo después de haber sido introducidos determinando que su solución sea más complicada y consuma más tiempo. Además no permite que se tenga un software desplegable en todo momento.

Como se desarrollan varios componentes a la vez y no se tiene definido claramente un orden de desarrollo y dependencia se presentan situaciones de bloqueo, que significa que un componente no funcione porque necesite algún servicio de otro componente que tampoco funcione porque necesita un servicio de él.

Una violación frecuente en el proyecto, es la del mismo concepto de componente. Un componente debe relacionarse con el resto sólo a través de su interfaz para que sea lo más independiente posible, lo que implica que debe ser transparente la implementación de un componente para el resto, algo que no se cumple. Es frecuente que se acceda a métodos de componentes incorrectamente y que se asuman datos directos de algún componente, en

vez de hacerlo mediante servicios. Esta situación crea una gran dependencia entre los componentes, provocando que sea muy complicado usarlos en otros contextos, o cuando existen cambios en uno, repercute significativamente en el resto.

Las pruebas para determinar el correcto funcionamiento de la integración no están definidas oficialmente por lo que la ocurrencia de defectos en las últimas etapas del desarrollo son habituales. La solución de estos errores eleva los costos del proyecto y aumentan el tiempo de desarrollo porque es más complicada su solución.

Todos los elementos expuestos tanto los organizativos como de violaciones de los lineamientos del desarrollo por componentes producen desviaciones en los plazos de entrega del sistema, elevan los costos del proyecto y provocan malestar en los integrantes. Muestra de ello se pudo comprobar en entrevistas realizadas a importantes roles (desarrolladores, arquitectos, jefe de proyectos, jefe de líneas) en los proyectos. Todos los entrevistados señalaron que han tenido desviaciones en sus respectivos proyectos por deficiencias en la gestión de la integración, pues el proceso no está organizado. Entre las principales causas expuestas están: demora en la implementación de servicios necesitados, no se planifican las tareas de integración, las personas que desarrollan la integración no tienen las competencias requeridas ni el tiempo necesario, no se le da seguimiento a las tareas de integración, entre otras causas. La entrevista realizada se encuentra en el Anexo 1.

Por todo lo antes expuesto se determinó el siguiente **problema científico de investigación**:

La deficiente gestión de la integración influye negativamente en la eficiencia del proceso de desarrollo de software en los proyectos del sistema CEDRUX.

Objeto de estudio: Proceso de desarrollo de software en CEDRUX.

Objetivo: Desarrollar y aplicar una guía metodológica para gestionar la integración de componentes, que contribuya a mejorar la eficiencia en el proceso de desarrollo de software en los proyectos del sistema CEDRUX.

Objetivos Específicos:

- Determinar el marco teórico de la investigación que permita conocer las principales tendencias en cuanto a la gestión de la integración en los proyectos de software en el mundo y los principales autores del tema.
- Desarrollar una guía metodológica para gestionar la integración de componentes en los proyectos del sistema CEDRUX.
- Aplicar la guía metodológica desarrollada en los proyectos del sistema CEDRUX y evaluar los resultados.

Campo de Acción: Proceso de Integración en el desarrollo de los proyectos del sistema CEDRUX.

Hipótesis: El desarrollo de una guía metodológica para gestionar la integración de componentes contribuirá a mejorar la eficiencia del proceso de desarrollo de software en los proyectos del sistema CEDRUX.

Tareas de Investigación:

1. Recopilación de la bibliografía necesaria para el estudio del desarrollo de software por componentes.
2. Análisis de las soluciones propuestas en las diferentes metodologías para gestionar la integración de componentes.
3. Preparación de las entrevistas para dimensionar el problema y aplicarla en los proyectos.
4. Aplicación de las entrevistas en los proyectos para dimensionar el problema.
5. Análisis de los criterios más comunes obtenidos en las entrevistas.
6. Elaborar la guía metodológica a seguir para la integración de componentes en los proyectos del sistema CEDRUX.
7. Ejecución del experimento en los proyectos del sistema CEDRUX.
8. Documentación de los resultados de la aplicación de la guía de acciones en los proyectos del sistema CEDRUX.
9. Preparación de la encuesta para evaluar la propuesta.

10. Aplicación la encuesta para evaluar la propuesta.
11. Documentación de los resultados del resultado de la encuesta para evaluar la propuesta.

Métodos y técnicas utilizadas:

Métodos Teóricos:

Analítico – Sintético: Después de hacer un análisis de los referentes teóricos y la bibliografía en cuestión se hará una síntesis de los elementos significativos de la investigación.

Inductivo – Deductivo: En este método a partir de la hipótesis y siguiendo reglas lógicas de deducción se llega a nuevos conocimientos y predicciones, las que posteriormente son sometidas a verificaciones empíricas (González 2002).

Histórico – Lógico: Este método se utilizó porque para resolver el problema de la investigación fue necesario estudiar la evolución que han tenido los conceptos más importantes que se tratan en la investigación como es el caso del desarrollo de software, la integración, entre otros. Por otra parte se estudió otras experiencias con puntos de coincidencia con la que aquí se propone.

Métodos Empíricos:

El experimento es el método empírico para el estudio de un objeto en el cual el investigador crea las condiciones o adapta las existentes para el esclarecimiento de las propiedades, leyes y relaciones del objeto, para verificar una hipótesis, una teoría o un modelo (González 2002).

En este caso se utilizó el método para determinar el efecto que causa la aplicación de la guía propuesta en el proceso de desarrollo de software en los proyectos.

Medición: Este método se utilizó para obtener información numérica acerca de una propiedad o cualidad del objeto, donde se comparan magnitudes medibles y conocidas (González 2002). Se utiliza para medir los indicadores definidos en la aplicación de la guía propuesta y en los definidos para la evaluación de la guía.

Métodos particulares

Como métodos particulares se utilizaron la entrevista para dimensionar el problema y la encuesta para evaluar la guía propuesta.

Aporte Práctico

La propuesta realizada tiene un importante aporte práctico y su aplicación puede contribuir a la mejora del proceso de desarrollo en los proyectos del sistema CEDRUX. Como parte de la investigación se obtienen los siguientes resultados:

- Guía metodológica para integración de componentes en los proyectos del sistema CEDRUX.
- Libro de procesos del área de procesos de integración.
- Herramientas de apoyo al proceso de integración
- Reglas para la generación automatizada de alertas y detección de riesgos integradas a la herramienta Redmine.

Estructura del trabajo:

Capítulo1: Se realiza un estudio del estado del arte y se analizan algunos temas como: modelos de desarrollo, estilos arquitectónicos, metodologías de desarrollo de software, entre otros temas. Se hace énfasis en el análisis de cada uno de los temas como se relaciona con la integración.

Capítulo 2: Se describe la solución propuesta.

Capítulo 3: Se hace un análisis de los resultados obtenidos en la aplicación de la propuesta en los proyectos del sistema CEDRUX. Se realiza además una evaluación de la propuesta.

Capítulo 1 Fundamentación Teórica

1.1. Introducción

En este capítulo se realizará una breve descripción de algunos modelos de desarrollo y lo que definen para la integración, se hará énfasis en el modelo de desarrollo por componentes y lo que está definido hasta el momento para la integración de componentes. Se analizará el estilo arquitectónico de desarrollo de software por componentes y algunas de sus particularidades que tienen alto impacto en el modelo de desarrollo. También se abordarán algunas de las metodologías de desarrollo de software profundizando en lo referente a la integración de componentes. Además se analizan otros temas vinculados con el desarrollo de software y la integración de componentes.

1.2. Estilos arquitectónicos

Mary Shaw y Paul Clements en (Clements 1996) definen los estilos arquitectónicos como un conjunto de reglas de diseño que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales. Según (Kiccillof 2005) es en gran medida la interacción entre los componentes, mediados por conectores, lo que confiere a los distintos estilos sus características distintivas. En la misma fuente se explican algunos de los estilos más utilizados, entre los que se encuentran: Arquitecturas en Capas, Arquitecturas Orientadas a Objetos, Arquitecturas Basadas en Componentes, Arquitectura de Máquinas Virtuales, Sistemas de control de procesos, Estilos Peer-to-Peer, Arquitecturas Basadas en Eventos, Arquitecturas Orientadas a Servicios, Arquitecturas Basadas en Recursos, Estilos de Flujo de Datos, Tubería y filtros, entre otros.

1.2.1. Arquitecturas Basadas en Componentes

Los sistemas de software basados en componentes se sustentan en principios definidos por una ingeniería de software específica: Component Based Software Component (CBSE) (Wallnau 1998). En (Kiccillof 2005) se enuncia que en un estilo de este tipo:

- Los componentes en el sentido estilístico son componentes en el sentido de CBSE y son las unidades de modelado, diseño e implementación.

- Las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

En la definición de estilo arquitectónico presentada se plantea que la interacción entre los componentes, mediados por conectores es lo que confiere a los distintos estilos sus características distintivas. Significa que en todos los estilos existen componentes, por ejemplo en el estilo por capas, los componentes son las diferentes capas. En el estilo que se está analizando esos componentes son precisamente componentes, por lo que es necesario diferenciar los conceptos de componente, el primero se refiere al sentido estilístico y el segundo se refiere a los componentes cuya definición se analiza en el siguiente epígrafe.

1.2.2. Componente

El término componente procede de las técnicas orientadas a objetos, de los problemas de descomposición usados en Técnicas de descomposición de problemas, y de su necesidad para desarrollar sistemas abiertos (Martínez 2003).

Para el Instituto de Ingeniería de Software (SEI, por sus siglas en inglés) un componente de software es un fragmento de un sistema de software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas (SEI 2008).

En (Martínez 2003) se realiza un análisis de esta definición y se plantea que se basa en tres perspectivas: (a) la perspectiva de empaquetamiento; (b) la perspectiva de servicio y (c) la perspectiva de integridad

En esta investigación se adopta como definición de componente la de Clemens Szyperski: Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que puede ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski 1998).

La idea de que el componente sea una unidad de composición, implica que no necesariamente tiene que ser construido durante el desarrollo, sino que puede ser variado su origen. En (Montilva, Arapé et al. 2005) se presentan algunas características de los componentes, que complementan esta definición:

- I**dentificable**: debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.
- **Accesible sólo a través de su interfaz**: debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación.
- **Servicios Invariantes**: las operaciones que ofrece a través de su interfaz no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.
- **Documentado**: debe tener una documentación adecuada que facilite su evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

1.2.3. Integración

En (Larsson 2005) se expresa que la integración representa el proceso desarrollado cuando las partes son combinadas para formar partes más complejas y finalmente en productos completos. Los elementos críticos en la integración incluyen descripción y gestión de interfaces, la secuencia en la cual los componentes son integrados y la comunicación entre diferentes involucrados. Incluso se reconoce que también se incluyen requerimientos y propiedades del sistema que no pueden ser comprobados desde un nivel de componente, pero que deben ser comprobados a nivel de sistema.

En (Duvall 2007) se expresa como el acto de combinar componentes de software (programas y archivos) en un sistema de software. En proyectos grandes se puede hablar de múltiples componentes pero que pueden llegar a ser sólo archivos de código de bajo nivel compilados para pequeños proyectos.

En ambas definiciones se coincide que el resultado de la integración es unir las partes por las que está constituido un sistema. Pero en el caso de la primera definición se reconoce explícitamente los elementos críticos en el proceso de integración, es decir se plantea como un proceso más complejo y no como una simple actividad. En este documento se adopta como concepto de integración la primera definición.

1.2.4. Consideraciones generales sobre los estilos arquitectónicos

Por el concepto de estilo arquitectónico presentado se puede confirmar que como todos los sistemas están compuestos por componentes (en el sentido estilístico) será necesario siempre integrarlos, por ejemplo en el estilo

por capas será necesario integrar las capas o en el estilo basado en componentes será necesario integrar los componentes. Además a la hora de identificar y desarrollar un componente debe asegurase que cumpla con las características especificadas en esta sección.

1.3. Modelos de desarrollo de software

Un proceso de software es un conjunto de actividades que conducen a la creación de un producto software. Estas actividades pueden consistir en el desarrollo de un software desde cero en un lenguaje de programación como Java o C. Cada vez más, se desarrolla nuevo software ampliando y modificando sistemas existentes y configurando e integrando software comercial o componentes. Un modelo de software es una representación abstracta de un proceso de software (Sommerville 2005).

1.3.1. Modelo en Cascada

El modelo en cascada o secuencial se utiliza cuando se tienen los requerimientos del sistema bien identificados, descritos, validados y sea improbable que cambien radicalmente durante el período de desarrollo. Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas. Las principales etapas de este modelo se transforman en actividades fundamentales para el desarrollo, estas son: análisis y definición de requerimientos, diseño del sistema y del software, Implementación y pruebas de unidades, Integración y pruebas del sistema, funcionamiento y mantenimiento (Sommerville 2005).

Respecto a la integración del sistema en este modelo, los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Destacando que como es secuencial el desarrollo, sólo se integrará luego que se halla implementado completamente las unidades independientes.

Para las condiciones del proyecto, la ejecución del modelo en cascada no es recomendable. La complejidad de la solución determina que por cada fase sea necesario dedicarle bastante tiempo, por lo que no se podría terminar una fase para comenzar la próxima. Un punto que tiene elementos positivos y negativos es lo definido en el modelo para las pruebas, que se conoce como modelo V, donde define para cada etapa la forma de probarlo. Lo positivo, es la planificación de pruebas de integración que demuestra que se le da importancia al tema, para validar el correcto funcionamiento de la colaboración entre los módulos y componentes que constituyen el sistema. Lo negativo, es

que realizar la integración de componentes después de haberlos desarrollado completamente, en la práctica trae muchos perjuicios pues cuando se realicen las pruebas se detectarán errores introducidos mucho tiempo atrás, por lo que su solución es más costosa. Por otra parte, cuando comience la integración de los componentes ya implementados, la realización de cambios es más compleja que durante su implementación.

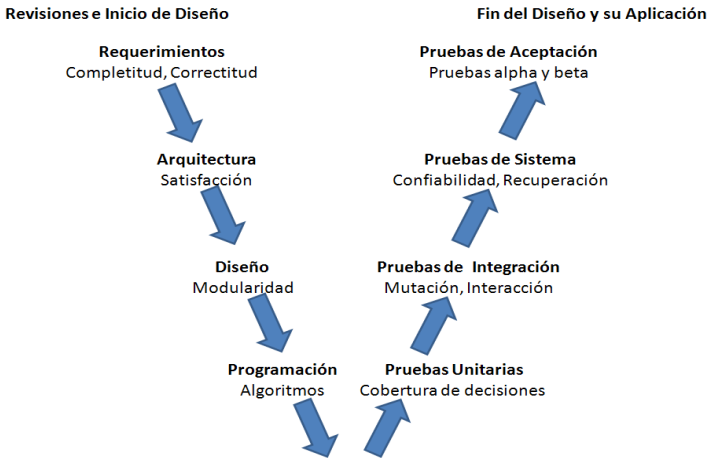


Figura 2 Modelo V

1.3.2. Modelos evolutivos

Se basan en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de diferentes versiones hasta que se desarrolla un sistema adecuado (Sommerville 2005). A continuación se caracterizan las principales tendencias de este modelo.

Modelo Incremental: Combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce “un incremento” del software. Por ejemplo, el software de tratamiento de textos desarrollado con el paradigma incremental podría extraer funciones de gestión de archivos básicos y de producción de documentos en el primer incremento, funciones de edición más sofisticadas y de producción de documentos en el segundo incremento, corrección ortográfica y gramatical en el tercero (Pressman 2001). El primer incremento a menudo es un producto esencial que afronta requisitos básicos. Este modelo se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son

versiones "incompletas" del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

Modelo en espiral: Conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Con este modelo el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado. (Pressman 2001).

Para el desarrollo de CEDUX sería beneficioso que se utilice un enfoque evolutivo. Por el tamaño del sistema, se puede desarrollar en los primeros momentos los subsistemas que constituyen la base y luego incorporar el resto de los subsistemas. Para cada subsistema se podrá realizar varias iteraciones, que culminarían con una versión cada vez más completa. Una de las ventajas que tiene el proyecto es que los clientes se encuentran permanentemente vinculados al desarrollo, lo que permite que si se desarrolla de forma evolutiva los clientes podrán revisar la evolución del sistema, facilitando su aceptación y la identificación de cambios en etapas tempranas del desarrollo.

1.3.3. Modelo de Desarrollo Basado en Componentes

Sommerville (Sommerville 2005) define a la Ingeniería de Software Basada en Componentes como "el proceso de definir, implementar e integrar o componer en sistemas independientes débilmente acoplados" Clasificando el desarrollo de software basado en componentes en dos categorías, la primera es la fabricación de componentes y la segunda es la construcción de software a partir del ensamblaje de componentes que han sido implementados. El desarrollo de componentes centra su principal esfuerzo en la construcción de funcionalidades integradas que en conjunto dan solución a una problemática existente en un domino determinado. Este proceso consiste en la adaptación o desarrollo de componentes, con el objetivo de ser reutilizados en futuras aplicaciones de dominios similares. Su principio es producir repositorios de activos que puedan ser reutilizados en el desarrollo de productos software. El desarrollo de software reutilizando componentes es un proceso en el cual el desarrollo de un nuevo producto involucra la reutilización de un conjunto de componentes existentes en repositorios, que sean confiables y que respondan de acuerdo a sus especificaciones según el dominio de su naturaleza.

En la etapa actual del desarrollo de CEDRUX se aplica este modelo en las dos variantes que define Sommerville, la primera es desarrollar los componentes y luego ensamblarlos para que conformen el sistema. Quizás lo más significativo es que se debe desarrollar los componentes pensando no sólo para el sistema actual, sino para que formen parte del repositorio de componentes de la organización y que puedan ser usados en el futuro para el desarrollo de otros sistemas. Para lograr ese objetivo los componentes deben cumplir con las características definidas en el epígrafe (1.3.2).

1.3.4. Líneas de Producto

En (Clements 2002) se define como líneas de productos de software (LPS) “un conjunto de sistemas de software que comparten un conjunto común y gestionado de aspectos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales de software de forma pre-escrita”. Para (Krueger 2010) las LPS "se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software, usando un medio común de producción".

En ambas definiciones se reconoce que todo parte de la posesión de activos reutilizables de software (requisitos, diseños, componentes, servicios web, etc.), luego el enfoque de LPS brinda una serie de técnicas para convertirlo en productos comunes utilizando el mismo medio de producción. En la Figura 3 se puede apreciar lo que se define en (Krueger 2010) como modelo básico de una LPS.

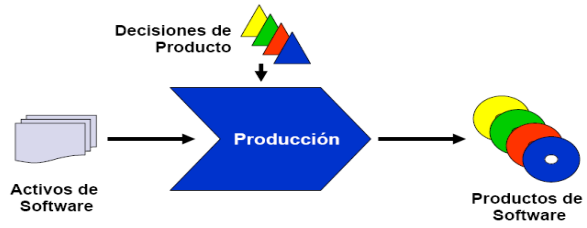


Figura 3 Modelo básico de una línea de producto

- La entrada: Activos de software. Una colección de partes de software (requisitos, diseños, componentes, casos de prueba, etc.) que se configuran y componen de forma pre-escrita para producir los productos de la línea.

- El control: Incluye los modelos de decisiones que describen los aspectos variables y opcionales de los productos de la línea. Además cada producto de la línea es definido por un conjunto de decisiones que se les conoce como decisiones del producto.
- El proceso de producción: Establece los mecanismos o pasos para componer y configurar productos a partir de los activos de entrada. Las decisiones del producto se usan para determinar qué decisiones de activos de entrada utilizar y cómo configurar los puntos de variación de esos activos.
- La salida**:** Conjunto de todos los productos que pueden ser o son producidos por la línea de productos.

En (Montilva 2006) se enuncian algunas de las ventajas de las LPS, entre las que se encuentran:

- La entrega de productos de software de una manera más rápida, económica y con una mejor calidad.
- Las LPS producen mejoras en: tiempo de entrega del producto, costos de ingeniería, tamaño del portafolio de productos, reducción de las tasas de defectos y calidad de los productos.
- Beneficios tácticos de ingeniería: reducción en el tiempo promedio de creación y entrega de nuevos productos, en el número promedio de defectos por producto, en el esfuerzo promedio requerido para desarrollar y mantener los productos, en el costo promedio de producción de los productos e incremento en el número total de productos que pueden ser efectivamente desplegados y mantenidos.
- Beneficios estratégicos de negocios: reducción en el tiempo de entrega y el tiempo de retorno de nuevos productos, mejoras en el valor competitivo del producto, márgenes mayores de ganancias, mejor calidad de los productos, mejoras en la reputación de la empresa, mayor escalabilidad del modelo de negocio en términos de productos y mercados, mayor agilidad para expandir el negocio a nuevos mercados y reducción de riesgos en la entrega de productos.

El tema de la reutilización es la base de este modelo de desarrollo. Como sugiere la Figura 4 el desarrollo por LPS es el escalón que le sucede al desarrollo basado en componentes. El proyecto para el desarrollo de CEDRUX tiene una estructura organizativa que favorece el tránsito del actual modelo al de LPS, pero todavía no se cuenta con la madurez necesaria en los activos de software en cada una de las líneas del proyecto, no obstante esta transición debe ser natural, en el momento en que los activos desarrollados cumplan con las condiciones. Con ese fin es

importante que los componentes sean lo más genéricos posibles pensando en su reutilización, para ello deben seguir los lineamientos expuestos en el epígrafe dedicado a los componentes (1.3).

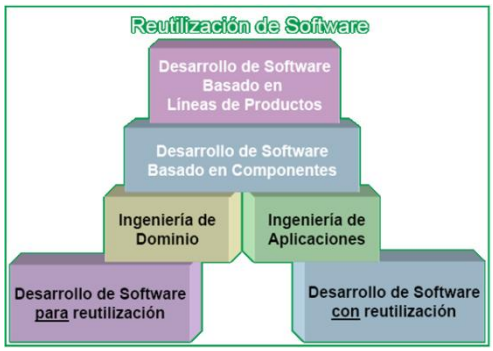


Figura 4 Evolución de la reutilización de software

1.3.5. Consideraciones sobre los modelos de desarrollo

La descripción de los modelos realizada permite hacer comparaciones entre ellos, pero lo más importante es conocer que en la actualidad la tendencia es a fusionarlos, por ejemplo en el modelo de desarrollo por componentes están presentes varios modelos como pueden ser los evolutivos. A partir del desarrollo por componentes han surgido otros modelos como las líneas de productos o las factorías de software. La integración adquiere relevancia en el desarrollo basado en componentes, debido a que estos se desarrollan de forma independiente y para que el sistema funcione tienen que estar integrados adecuadamente; en los sistemas evolutivos también la integración se ve presente durante todo el proceso de desarrollo; mientras que en el modelo en cascada es menos complejo porque es al final de la implementación que se comenzará a integrar, lo que produce grandes desventajas.

Por las características del proyecto y la organización que lo ejecuta, el modelo de desarrollo por componentes es el que se aplica, se desarrollará el sistema y cada componente de forma evolutiva, aplicando los elementos que este modelo define. Esta determinación genera importantes ventajas mencionadas anteriormente, entre las que se encuentra la posibilidad de contar en el futuro con un repositorio de componentes y que los funcionales vayan revisando versiones intermedias del sistema.

1.4. Metodologías

Para hacer un análisis de las metodologías de desarrollo de software lo primero que se debería hacer es analizar qué significa el término metodología. Con ese fin es provechoso lo expuesto en (Cockburn 2001), donde el autor expresa su aceptación con lo definido en el diccionario Merriam-Webster como metodología, “una serie de métodos o técnicas relacionadas” y que un método es “un procedimiento sistemático ” similar a una “técnica”. El autor también analiza la diferencia con la definición del diccionario Oxford English Dictionary donde se define metodología como “el estudio de métodos”. La variedad de definiciones contribuye a la controversia sobre el término metodología. La distinción entre metodología y método también es importante, en la misma fuente el autor señala como ejemplo las habituales frases de “Un método para encontrar clases desde los casos de uso” o “diferentes métodos son usados para resolver diferentes problemas”, entendiendo que se habla de técnicas o de procedimientos, sin establecer reglas y convenios para el equipo. Con ese enfoque el uso de la palabra metodología queda para el propósito superior de coordinar las actividades de las personas en un equipo.

Sin embargo en (Bustos 2003) no se diferencia el término método de metodología y plantea que: “Un método, comúnmente llamado metodología, impone un proceso disciplinado sobre el desarrollo de software con el objetivo de hacer el desarrollo de software más predecible y eficiente. Por tanto, se plantea que un método define un camino reproducible para obtener resultados confiables”. El autor también plantea que “un método de desarrollo de software describe cómo modelar y construir un sistema de software de una forma confiable y reproducible.” y que “Los métodos también definen una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas. Una buena representación busca un balance entre la densidad de información y la legibilidad. En relación con los elementos del modelo y su representación gráfica, un método define las reglas que describen la resolución de los diferentes puntos de vista, el orden de las tareas y la asignación de responsabilidades. Estas reglas definen un proceso que asegura armonía dentro de un grupo de elementos cooperativos y explican cómo debería ser usado el método.”

En las dos fuentes hay similitud en la esencia del significado de metodología, quizás lo más relevante es el uso del término método en la segunda fuente, con el mismo significado de metodología, en la primera fuente se aclara bien la diferencia entre esos términos. No obstante, en la segunda definición se brindan elementos que ayudan a entender la función de una metodología y sobresalen términos como eficiencia, predecible, reproducible,

legibilidad, armonía y elementos cooperativos. En (Cockburn 2001) se enuncian los elementos que constituyen una metodología como se muestra en la Figura 5, estos elementos son: roles, competencias, equipos, técnicas, actividades, procesos, artefactos (productos del trabajo), hitos, estándares, calidad, valores del equipo.

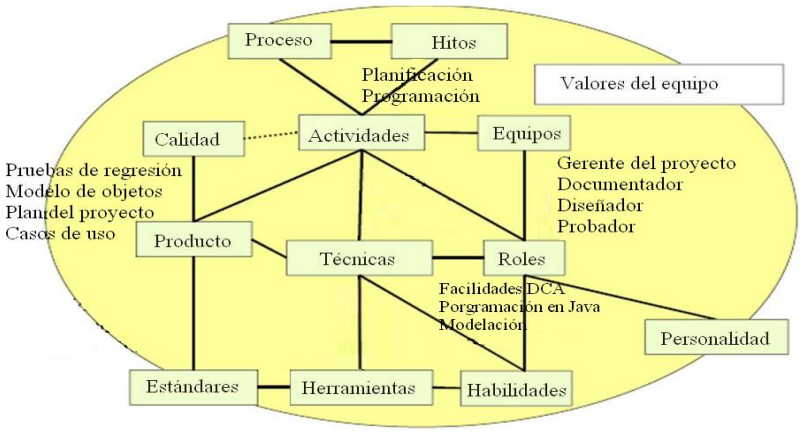


Figura 5 Elementos de una metodología

Las metodologías en la actualidad se dividen en dos grandes grupos, las llamadas metodologías tradicionales o pesadas y las metodologías ágiles. Dentro de las tradicionales la mayor exponente es RUP. Dentro de las ágiles se pueden mencionar XP, SCRUM, CRYSTAL, entre otras.

Las ágiles surgen como alternativa a las metodologías tradicionales intentando establecer un justo equilibro entre "sin proceso" y "demasiado proceso", proporcionan sólo el proceso suficiente para obtener un retorno razonable. De muchas formas, estos métodos están más orientados al código: siguiendo la idea de que la parte clave de la documentación es el código fuente (Bustos 2003). En (Beck, Beedle et al. 2001) se enuncia el manifiesto ágil, el cual plantea lo siguiente :

- Los individuos y sus interacciones, son más importantes que los procesos y herramientas.
- Un software que funcione, es más importante que una abundante documentación.
- La colaboración con los clientes, es más importante que la negociación de contratos.
- La respuesta ante el cambio, es más importante que el seguimiento de un plan.

Además del manifiesto el desarrollo ágil se rige por doce principios que se enuncian en la misma fuente:

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

A continuación se analizan las características de algunas metodologías, en el caso de las tradicionales se analiza RUP, en el caso de las ágiles XP, que es las que más conocida según en estudio publicado en (Dybá 2008) el 76 % de los trabajos publicados sobre métodos ágiles con un enfoque práctico correspondían a XP. Se analiza Microsoft Solution Framework (MSF) por ser la metodología utilizada por la empresa Microsoft una de las más destacadas en el desarrollo de software.

1.4.1. Rational Unified Process (RUP)

RUP es quizás la metodología más utilizada en las empresas desarrolladoras de software, su desarrollo se divide en cuatro fases y nueve flujos de trabajo, dentro de los que se encuentra el flujo de trabajo implementación. En el cual se definen un conjunto de actividades encaminadas a la construcción e integración de los componentes de un sistema, estas se describen a continuación (Rational 2003):

- Estructurar el modelo de implementación: Generalmente la confección del modelo de implementación se realiza a partir de un conjunto de subsistemas de implementación que pueden ser desarrollados relativamente independientes. Es una actividad regida por el arquitecto en la Fase de Elaboración, permite organizar el modelo de implementación para administrar su complejidad, y se reconocen las funcionalidades principales que conforman el sistema, desde el punto de vista físico.

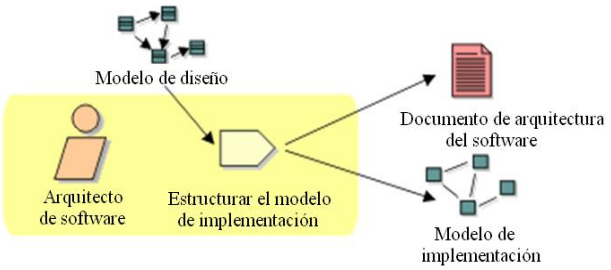
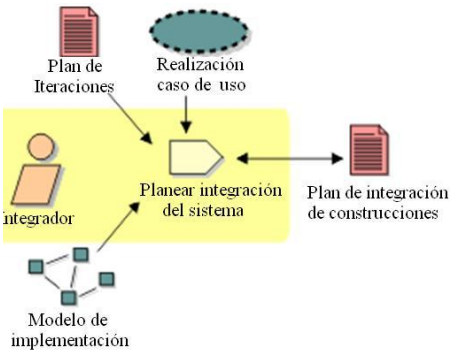


Figura 6 Estructurar el modelo de implementación

- Planear la integración: Se debe elaborar un plan que contenga la planificación de la integración, el cual estará enfocado en los subsistemas de implementación que se implementarán y el orden en que los subsistemas de implementación deben ser integrados en la iteración actual.



- Implementar componentes: Se detallan los elementos que conformarán un componente y se desarrollan cumpliendo los requisitos del sistema.
- Integrar componentes o subsistemas desarrollados: Aquí es donde se integran los componentes que ya han sido implementados y validados según la planificación y los mecanismos detallados en el Plan de Integración. Se describen los cambios ocurridos en la integración por varios programadores para crear una nueva versión consistente del componente.

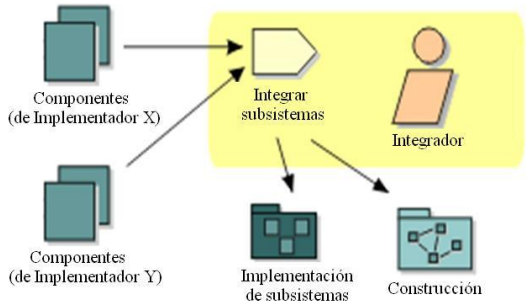


Figura 8 Integrar componentes o subsistemas

- Integración de componentes al sistema: El integrador, guiándose por el Plan de Integración de Construcciones, va adicionando los componentes liberados para la integración dentro del área de trabajo de integración de sistemas. En cada construcción la integración es probada. Después de la última integración a la construcción se le pueden realizar pruebas en el sistema completo.

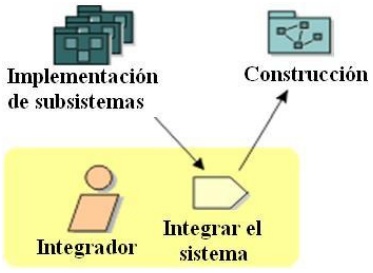


Figura 9 Integración de componentes al sistema

Además de las actividades específicas para la integración en RUP se definen artefactos, los cuales son responsabilidad del rol del integrador como se puede apreciar en la Figura 10.

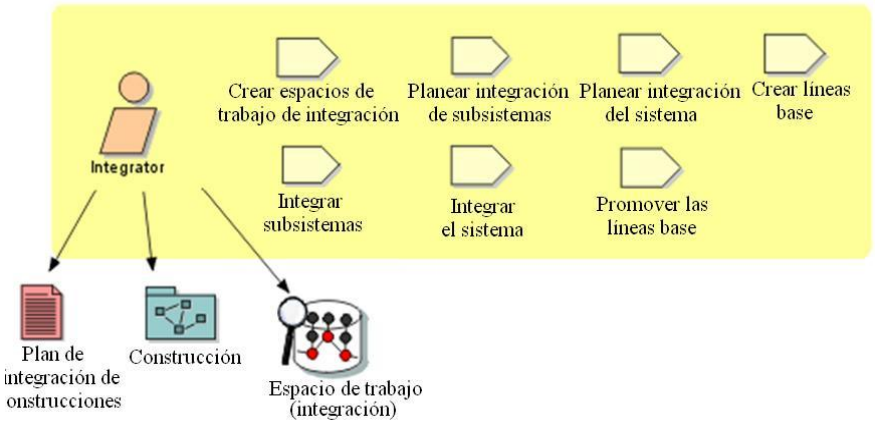


Figura 10 Responsabilidades del integrador

En RUP, se reconoce la importancia que tiene la integración, por ello se definen roles específicos que sean los principales responsables del ensamblaje de las diferentes partes que componen el sistema. Además se especifican las actividades relacionadas con la integración y cuales son los artefactos que se generan de su realización. Otro elemento, hasta cierto punto positivo es que se planifican pruebas de integración para demostrar que el sistema funciona correctamente después de integrar sus componentes.

Pese a la importancia reconocida en RUP a la integración, existen definiciones que tienen un impacto negativo en la calidad del producto de software y en su proceso de desarrollo. Quizás la decisión que más afecta el correcto proceso de desarrollo es la que dicta que la integración se realiza luego que se hallan implementado completamente los componentes, es decir al final de la iteración. Uno de los problemas que trae esta regla es que en la integración se detectarán errores introducidos mucho tiempo atrás, por lo que los desarrolladores les será más complicada su solución y tendrán que dedicar más tiempo que si se hubiese detectado antes. Otro elemento no menos significativo es que los clientes tendrán que esperar el flujo de la integración para comprobar si el sistema funciona como ellos esperan, sólo en ese momento podrán apreciar la interacción y el intercambio de datos e informaciones entre los subsistemas.

Un elemento negativo es que como para la integración se sigue un plan, puede que se detecten muchos errores y su solución consuma más tiempo del planificado, en esos casos se suele tomar la decisión de dejar la integración para próximas iteraciones. Una de las particularidades del desarrollo por componentes es lo que se conoce como composición tardía, lo que significa que se pueden ir incorporando nuevos componentes al sistema paulatinamente. En RUP este elemento puede ser un verdadero problema porque define que cuando se termina y está probado un componente del sistema pasa a ser un elemento de configuración del software, miembro de la línea base del proyecto, si algunos de los componentes que están en desarrollo necesita un servicio del componente terminado y este no lo brinda se crea una situación que pueda llevar mucho tiempo porque es necesario pasar por un proceso de solicitud de cambio riguroso, lo que puede implicar una extensión significativa del tiempo de desarrollo. La situación antes descrita es muy frecuente en el desarrollo de CEDRUX y debido a que se debe tener un resultado importante en poco tiempo, no se puede seguir con este lineamiento algo "burocrático" de RUP.

1.4.2. Extreme Programing (XP)

A diferencia de las metodologías pesadas, donde se ve el desarrollo de software como un proceso industrial, en las ágiles se ve como un proceso social, donde intervienen personas, que de su creatividad y características personales y colectivas dependerá el resultado del futuro sistema. Este enfoque tuvo como su principal precursor a Frederick Brooks con su clásico libro "El mítico hombre mes"(Brooks 1995).

La programación Extrema (XP) como se plantea en (Beck 2009) por uno de los padres del desarrollo ágil es una metodología ágil, centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Los roles en XP de acuerdo con la propuesta original de Beck son: programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador, consultor y el gestor.

En (Canós 2005) se expone que la principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione.

Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas:

- El juego de la planificación. Hay una comunicación frecuente entre el cliente y los programadores.
- Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema.
- Metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.
- Diseño simple. Diseñar la solución lo más simple que pueda funcionar y ser implementada.
- Pruebas. La producción de código está dirigida por las pruebas unitarias.
- Refactorización: Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.
- Programación en parejas. La codificación se realiza con trabajo en parejas de programadores.
- Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código.
- Integración continua. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- 40 horas por semana. Se debe trabajar un máximo de 40 horas por semana.
- Cliente in-situ. El cliente tiene que estar presente y disponible todo el tiempo para el equipo.
- Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación.

Como se puede apreciar en las prácticas bases de XP la integración continua es una de ellas. A continuación se profundiza en este tema por la relevancia que posee para la investigación en curso.

1.4.2.1. Integración continua

Una de las características fundamentales en la metodología XP es la práctica de integración continua. Según uno de los principales exponentes del desarrollo ágil, Martín Fowler, una de las cosas más difíciles de expresar acerca de la integración continua es que hace un cambio fundamental en el patrón de desarrollo conjunto, además señala que el tema de la integración continua pone un poco de disciplina cuando se trabaja con equipos de muchas personas donde suelen surgir problemas relacionados con la integración (Fowler 2006). En XP se integra el nuevo código en la base de código existente varias veces al día, de no realizarse la velocidad del desarrollo se verá afectada y puede traer como resultado la utilización de días o incluso semanas, tratando de solucionar la acumulación de errores que surgieron durante las grandes integraciones.

Una de las ventajas que brinda esta práctica es la velocidad, porque al integrar continuamente se detectarán errores de integración en fases tempranas cuando los problemas aún son pequeños y fáciles de detectar, lo que le permitirá mantener la velocidad máxima todo el tiempo. Otra de las ventajas es la reducción de riesgos, un ejemplo que ayuda a ilustrar esto es que la integración tradicional y la continua son como la diferencia entre estar de pie bajo la lluvia por un par de horas y de pie bajo las cataratas del Niágara. Es decir que si se deja la integración como un evento único probablemente se sufra un gran atraso por lo que generalmente se aplazará la integración para iteraciones posteriores, repitiéndose en el futuro los mismos problemas, lo que determina que muchos proyectos de software se atrasen e incluso fracasen.

Otra ventaja importante es la identificación de problemas de integración. Como se realiza constantemente y se integra el nuevo código al código que ya está probado y estable, si existe algún problema es evidente que el error está en el nuevo código por lo que se puede identificar rápidamente su origen.

En caso de proyectos grandes es necesario usar para la integración continua como recomienda Fowler un sistema para la gestión de la configuración. En el sistema para la gestión de la configuración debe estar todo el código, además este tipo de sistema maneja fácilmente el versionado de cada archivo, es decir que se puede obtener versiones anteriores de cualquier archivo fácilmente. Fowler también reflexiona en (Fowler 2006) que se suele separar los proyectos para los diferentes componentes, lo que trae como problema que las personas tienen que recordar con qué versión de los otros componentes trabaja cada versión de cada componente.

Relacionado con la integración están las pruebas, en XP se definen dos tipos: las de unidad que son realizadas por los desarrolladores y las de aceptación que son realizadas por los clientes o un grupo externo y que cuentan con la ayuda de los desarrolladores. Para la realización de esas pruebas es recomendable el uso de herramientas automatizadas que agilicen el proceso y que den garantía de calidad. Estas prueban ayudan a garantizar que no se integre código incorrecto y que el sistema funcione correctamente.

En (Fleischer 2009) se realiza un estudio de las compañías que utilizan la integración continua y de las herramientas más utilizadas, entre las que se destacan: AnthillPro, Bamboo, Continuum, Cruise Control, FinalBuilder, Hudson, Lunt build, Parabuild, Pulse, Quick build, TeamCity. En esa investigación se revelaron datos muy interesantes, por ejemplo el 90,8% de las compañías estudiadas usan soluciones de integración continua y el 7,7% no la usa. La solución más utilizada fue Cruise Control seguido de Hudson. Los encuestados además señalaron entre las mayores ventajas de la integración continua la ejecución de las pruebas, la compilación del código, el análisis y cobertura del código y el análisis de la calidad del código. Ante la pregunta de si la integración continua había mejorado su proceso de desarrollo, el 77 % dio una respuesta positiva, 5 % acotaron que habían mejorado parcialmente, el otro 18 % no respondió la pregunta. De las personas que respondieron positivamente señalaron que la integración continua había reducido el tiempo en la detección de errores, que había mejorado notablemente la velocidad del desarrollo y que en general había mejorado la calidad de los proyectos de software. Por tales motivos el 77% de los encuestados manifestaron deseo de usar la integración continua en el futuro.

Se han publicado diversos artículos y libros sobre las ventajas y desventajas de la integración continua, el más referenciado es “Continuous Integration, Improving Software Quality and Reducing Risks” por Paul Duval, donde analiza el impacto sobre la calidad del software y la reducción de riesgos. En otro artículo se analizan por el mismo autor algunas de las malas prácticas a la hora de llevar a cabo la integración continua, a este fenómeno se le ha nombrado anti patrones, que según lo definen en (Duvall 2009) son soluciones que parecen ser beneficiosas, pero, al final tienden a producir el efecto adverso. Según el artículo, algunos de los anti patrones frecuentes en la aplicación de la integración continua son:

- Infrecuentes actualizaciones, genera atrasos. La solución es subir partes pequeñas de código frecuentemente, impedirá los dolores provocados en la tradicional gran integración.

- Construcciones rotas, significa que las construcciones se mantienes rotas por un largo período de tiempo lo que impide a los desarrolladores probar el funcionamiento del código, también impide a los equipos moverse a otras tareas. La solución es que se le notifique inmediatamente a los desarrolladores sobre la construcción rota y le den máxima prioridad para resolver el problema.
- Mínima retroalimentación, lo que impide que se produzca acción. Como solución se deben utilizar diversos mecanismos de retroalimentación lo que posibilitará a los desarrolladores conocer el estado de sus construcciones y en caso que existan fallos actuar rápidamente.
- Recepción de comentarios spam, lo que provoca que las personas ignoren los mensajes. La solución es garantizar que al equipo no le llegue información irrelevante.
- Poseer una máquina (servidor) lenta, lo que retrasa la retroalimentación. La solución es poseer una máquina con las prestaciones necesarias para que las construcciones se ejecuten con rapidez.

Sin dudas, la principal evidencia de la importancia que se le otorga en XP a la integración, es que se desarrolle continuamente y no se deje para el final. Las ventajas de este enfoque se explicaron en los epígrafes anteriores y según los estudios realizados las empresas que llevan a cabo esta práctica han tenido excelentes resultados. Como elemento débil en XP está el hecho que no se definen los procedimientos para gestionar la integración de componentes, no se especifican las actividades específicas para terminar los puntos de integración entre componentes y se le da toda la responsabilidad al desarrollador, que en caso de sistemas grandes representa una deficiencia. Igualmente no se especifican artefactos que sirvan de apoyo al proceso de integración, a la toma de decisiones y a la reutilización. Algunas de estas deficiencias están justificadas por el hecho de que para el trabajo en XP se debe tener un elevado nivel técnico y el equipo debe tener un alto nivel de comunicación.

1.4.3. Microsoft Solution Framework (MSF)

MSF es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF posee varios principios como: Promover comunicaciones abiertas, trabajar para una visión compartida, fortalecer los miembros del equipo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio, permanecer ágil y esperar los cambios, invertir en la calidad y aprender de todas las experiencias (Microsoft. 2003).

Según se expone en (Microsoft 2005) en MSF las acciones para la integración de las funcionalidades a desarrollar son ejecutadas por dos roles fundamentales: el desarrollador y el arquitecto, cada cual con sus tareas específicas. El desarrollador tiene como principal meta la implementación de la aplicación en los tiempos definidos. Para ello implementa los componentes añadiendo nuevas funcionalidades en diferentes iteraciones según la arquitectura del sistema. Define los puntos de integración en conjunto con los otros miembros del equipo de desarrollo para establecer una idea clara de cómo las funcionalidades serán integradas entre sí. Para implementar una nueva funcionalidad en un componente es necesario identificar aquellas áreas donde pueden producirse cambios y escribir las pruebas unitarias para validar ese código, su revisión y corrección, finalmente se integra el nuevo código en el servidor de control de versiones. Una vez desarrollado el componente debe ser probado, revisado y se debe analizar su código como una unidad íntegra, posteriormente se integra a la aplicación y se prueba la integración.

Para proporcionar un entorno estable y coherente en la aplicación, la integración del código debe ser organizada. El código se va integrando como pequeños elementos de cambios, representando una nueva funcionalidad o corrección de errores y manteniendo un nivel de estabilidad verificable. Se realizan las pruebas de unidad para asegurarse de que los cambios son compatibles con las funcionalidades ya integradas. Esto hace que sea más fácil de determinar y aislar un problema potencial. Cuando el componente final se integra, el desarrollador responsable prueba las funcionalidades implementadas de extremo a extremo y posteriormente se asigna a un probador.

El arquitecto por su parte tiene como actividad fundamental diseñar la arquitectura de la solución a desarrollar, para ello realiza un conjunto de actividades específicas entre las que se encuentra la división de la aplicación en componentes, identificar los flujos de datos entre componentes y subsistemas individuales, así como las acciones a realizar para la integración de los componentes. Debe garantizar que las definiciones de las interfaces tengan suficiente detalle como para que los desarrolladores puedan proceder con su diseño e implementación. Trabaja en conjunto con los desarrolladores en la definición de métodos en las clases que proporcionarán las interfaces de

comunicación entre los elementos desarrollados. Ordena las funcionalidades según la prioridad para facilitar la implementación y hacer que la integración sea más fluida. Utiliza además el diagrama de clases para comprobar la consistencia de las clases y los nuevos métodos que permitirán la integración. Ayuda a facilitar la integración de las funcionalidades comprobando que todos los mecanismos para la integración se entienden.

En el estudio realizado se pudo comprobar que aunque MSF tiene un enfoque ágil de desarrollo se definen actividades y roles específicos para desarrollar la integración. No obstante no se definen claramente los artefactos que sirven de apoyo al proceso de integración y que pueden ser importantes en el rediseño de soluciones y en la reutilización de componentes.

1.4.4. Consideraciones sobre las metodologías

En la sección dedicada a los estilos arquitectónicos se concluyó que independientemente del estilo arquitectónico que se emplee en el desarrollo de un sistema, siempre será necesario integrar debido a que el sistema estará compuesto por componentes (en el concepto estilístico) y conectores. Por este motivo, todas las metodologías deberían especificar elementos relacionados con la integración, como son: actividades, artefactos, roles y responsabilidades, entre otros. Se pudo identificar los principales enfoques con que se trata la integración, la primera tendencia es a desarrollar los componentes y sólo al final llevar a cabo la integración, a esta tendencia responden las metodologías tradicionales y definen una fase para llevar a cabo “la gran integración”. La segunda tendencia es a desarrollar la integración continua, desde el principio de la implementación, con este enfoque es que se trata el tema en las metodologías ágiles.

Según las investigaciones consultadas se pudo corroborar que la integración continua cuenta con gran aceptación entre los desarrolladores de software en el mundo y que los resultados de su aplicación han sido satisfactorios, además de brindar grandes ventajas, en detrimento de la tradicional. En el proyecto que se estudia, determinado por el tipo de sistema, en este caso un ERP, donde sus mayores beneficios está en la integración de todos los procesos importantes de una empresa, es necesario que se desarrolle con integración continua, pues de no ser así una fase de integración podría demorarse demasiado tiempo y es precisamente el tiempo una de las restricciones que tiene el proyecto, por otra parte no se podría asegurar que en una fase de integración, por el elevado número

de componentes y la alta relación entre ellos se puedan detectar y resolver todos los errores que aparezcan. A la hora de aplicar la integración continua deben tenerse en cuenta las buenas prácticas y anti patrones explicados.

Por las características del equipo de trabajo, que es inexperto en su mayoría y variable, es necesario adoptar algunos de los elementos definidos en RUP para llevar a cabo la integración, específicamente algunas actividades, roles y artefactos. Se debe tener en cuenta en la solución que se propondrá, características propias del desarrollo por componentes, dentro de las que se encuentra la composición tardía. Resulta muy interesante lo definido en MSF donde se establece la responsabilidad del desarrollo de la integración al arquitecto y al desarrollador y a cada uno se le establecen acciones específicas que tienen que desarrollar. Este enfoque es de mucha utilidad en el proyecto en que se enmarca esta investigación.

1.5. Herramientas de automatización de procesos y ayuda a la toma de decisiones

En la actualidad, prácticamente todas las industrias han automatizado buena parte de sus procesos, como por ejemplo la industria automovilística, donde la producción de carros se realiza prácticamente sin la intervención humana. Sin embargo, constituye una paradoja que en la industria de desarrollo de software esta tendencia no se comporta así. En (Duvall 2007) se compara esta situación con la de el zapatero que brinda zapatos a todos sus clientes y que olvida los zapatos de sus propios hijos. Además se reconoce que aunque el desarrollo de software generalmente es complejo existen actividades repetitivas propensas a errores que pueden ser automatizadas y se señala que cuando todo el trabajo se desarrolla manualmente queda muy poco tiempo para monitorear, controlar el seguimiento del proceso y del producto, planificar mejoras y así sucesivamente.

Tomando en consideración las ventajas de la automatización se deben utilizar herramientas automatizadas que contribuyan a mejorar la productividad del trabajo, que permitan detectar y reducir riesgos automáticamente y que brinden información constantemente de las irregularidades en el proceso. Se debe lograr algo similar a lo que ocurre en una fábrica que tiene los procesos automatizados, donde si una máquina tiene alguna dificultad se emite una alarma o si uno de los subproductos usados no cumple ciertos parámetros se activará algún mecanismo de alerta. Todas estas medidas ayudan a la toma de decisiones, pues se brinda constantemente información del desarrollo del proceso.

1.6. Experiencias en Cuba

No se encontraron publicaciones que muestren experiencias de cómo se gestiona la integración de componentes en las empresas de desarrollo de software en Cuba, lo que demuestra que el tema es subvalorado en una gran parte de los proyectos de software. Por este motivo para conocer como es tratado el tema se realizó una encuesta (Ver Anexo 3) en algunas de las empresas desarrolladoras de software en el país. Las empresas donde se realizó la encuesta fueron: Desoft en Guantánamo, Datys en Matanzas, Softel en la UCI y Transoft en la Habana.

Por lo general en todas se limitan a gestionar la integración con lo definido en la metodología de desarrollo que usan, por lo que se le presentan los problemas explicados en el análisis de las metodologías. En caso de Softel, donde usan RUP y además aplican el modelo de desarrollo por factorías de software, reconocen que frecuentemente sufren atrasos por problemas con la integración y manifiestan su impresión de que sería muy importante un procedimiento que guie el proceso de integración.

Algo similar sucede en la UCI, donde no se encontraron publicaciones de cómo es gestionada la integración en los proyectos de desarrollo que se ejecutan. Se aplicó la encuesta en tres centros, que sumados al CEIGE suman cuatro lo que representa el 30 % de los trece que se encuentran en la universidad. Los centros donde se realizó la encuesta fueron: Centro de Gobierno Electrónico (CEGEL), Centro de Gestión de Información y Tecnologías Libres (GEITEL) y el Centro de Informática Industrial (CEDIN). El caso más interesante se encontró en el centro GEITEL, donde usan la metodología de desarrollo SXP, que es la unión coherente de las metodologías XP y SCRUM y se detalla en (Rodríguez 2008). Los encuestados añaden que tuvieron frecuentes atrasos por esta causa en el pasado, pero desde que usan esa metodología han mejorado considerablemente; aplicando las prácticas de integración continua propuestas con XP y ayudados con los elementos definidos por SCRUM para mantener la comunicación constante de los equipos de desarrollo. Como es el caso de los encuentros que se desarrollan diariamente y contribuyen a que se traten directamente entre los desarrolladores lo referente a la integración. Resulta muy interesante lo que se plantea en este centro, el enfoque de realizar encuentros formales entre todo el equipo de desarrollo es muy positivo, aunque por la magnitud del equipo de desarrollo de CEDRUX esto no sería muy productivo, aunque si como parte de la propuesta se tendrá en cuenta los encuentros personales para acordar detalles de integración. Otro elemento a resaltar en este centro es la aplicación de lo que se conoce como Test de Joel (Ver Anexo 6), que

constituye una prueba con doce preguntas que ayudan a la retroalimentación constante del funcionamiento de los proyectos.

Otro elemento que se considera interesante es la organización que se asume en el CEDIN, donde existen equipos de desarrollo y equipos especializados en la integración. Este modelo teóricamente es bueno, pues promueve la especialización de los roles y el resultado debería ser satisfactorio, pero en la práctica no funciona tan bien, por lo menos si se aplica rigurosamente. Como los especialistas de integración son externos al equipo de desarrollo no tienen el conocimiento suficiente del negocio y de los detalles de la implementación que le permita desarrollar satisfactoriamente la integración. Por este motivo plantean que han ido adaptando el modelo con el tiempo y actualmente lo que hacen es mantener los equipos separados, donde los especialistas de integración se les prepara con todos los elementos teóricos del tema, pero cuando comienza un proyecto se relaciona con este desde el principio como un miembro más del equipo, donde sus conocimientos teóricos de integración y con los conocimientos sobre el negocio que le van aportando el resto del equipo del proyecto permite que la integración se desarrolle satisfactoriamente. Gracias a esta evolución plantean que han ido mejorando y reduciendo los atrasos por motivos de la integración.

En el caso de uno de los proyectos del CEGEL exponen que la integración le provocó atrasos significativos, pues aplicando lo definido en RUP la realizaron luego de terminar los módulos. Esto provocó que se tuvieran que realizar cambios significativos en algunos de estos pues se habían obviado elementos fundamentales que fueron detectados cuando comenzó la integración, al final tuvieron que desplegar en un inicio una versión del sistema que no estaba integrada completamente porque el proceso de integración se llevó mucho más tiempo del planificado. Esto demuestra las ventajas que tiene la integración continua.

1.7. Conclusiones del capítulo

- El estudio del concepto de estilo arquitectónico permitió conocer que independientemente del estilo que se utilice siempre será necesario integrar los componentes por los cuales está dividido el sistema. El estudio del concepto de componente sirve de guía para la identificación de componentes y su relación con el resto de los componentes del sistema. Dentro de las características más importantes que deben cumplir los componentes se encuentran que deben ser identificables, accesibles sólo a través de su interfaz, sus servicios deben ser invariantes y documentados.
- El estudio de algunas metodologías y modelos de desarrollo de software propició que se detallaran sus características, sobre todo cómo se trata el tema de la integración. Con este estudio se pudo determinar las dos principales tendencias con las que se ejecuta la integración de componentes, la primera es desarrollar la integración sólo cuando hallan sido implementados los componentes y la segunda es comenzar a integrar desde el inicio y durante todo el proceso de implementación. La primera tendencia es la que se propone en las metodologías tradicionales como RUP y la segunda es la propuesta por las metodologías ágiles. Además se pudo demostrar que por la características del proyecto es necesario efectuar la integración continua, no obstante se deben tener en cuenta elementos definidos en las metodologías tradicionales, entre los que se encuentran los roles, artefactos y actividades específicas para desarrollar la integración.
- La bibliografía consultada sobre los diferentes temas es abundante y actualizada lo que permite brindar una solución acorde a las mejores prácticas en el mundo del desarrollo de software, no obstante se debe señalar que el tema de la gestión de la integración es subvalorado, los principales estudios se enfocan generalmente en la parte tecnológica y obvian lo relacionado con el proceso de gestionar la integración, aunque la mayoría reconocen que cuando no se realiza organizadamente tiene un impacto negativo en el éxito del proyecto y la calidad del sistema. Esto se pudo evidenciar en la situación de Cuba, donde no se encontraron artículos vinculados con el tema, no obstante mediante otros mecanismos de búsqueda de información se pudo conocer algunas de las experiencias en algunos de los proyectos de Cuba, fundamentalmente en la UCI. Las experiencias más interesantes resultaron lo aplicado en el centro GEITEL y CEDIN.

Capítulo 2 Descripción de la guía metodológica

2.1. Introducción

En este capítulo se realiza una explicación de los elementos constituyentes de la propuesta. En su realización se tuvo en cuenta lo definido en el marco teórico y las ideas señaladas en las entrevistas realizadas a importantes roles de los proyectos.

2.2. Estándar para la integración

Para la integración entre subsistemas se usa como mediador un fichero xml nombrado IOC (Inversion of Control), donde cada subsistema publica sus servicios, en la Figura 11 se muestra esta organización. En el caso de la integración entre los componentes internos de un proyecto se usa algo similar solo que en ese caso el mediador es un xml nombrado IOC interno, cada subsistema posee uno para que sus componentes publiquen los servicios que le brindan al resto de los componentes del proyecto, en la Figura 12 se muestra esta definición para la integración interna. En el caso de la integración con otros sistemas se implementa una interfaz donde se ubican todos los servicios que brinda CEDRUX, en la Figura 11 se puede apreciar esta definición.

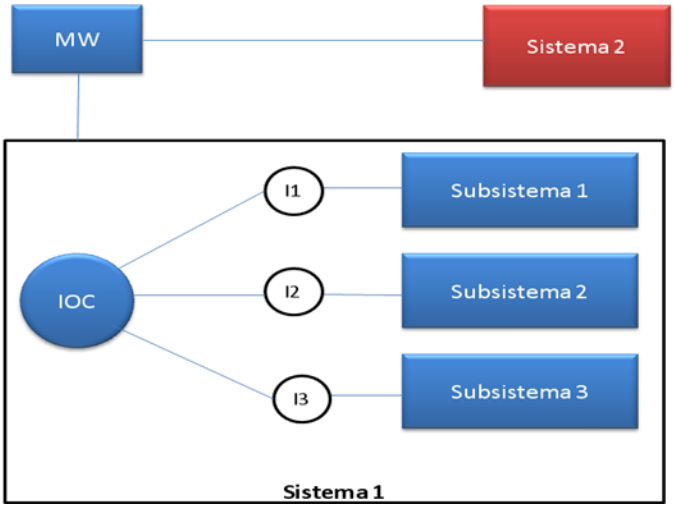


Figura 11 Integración entre subsistemas

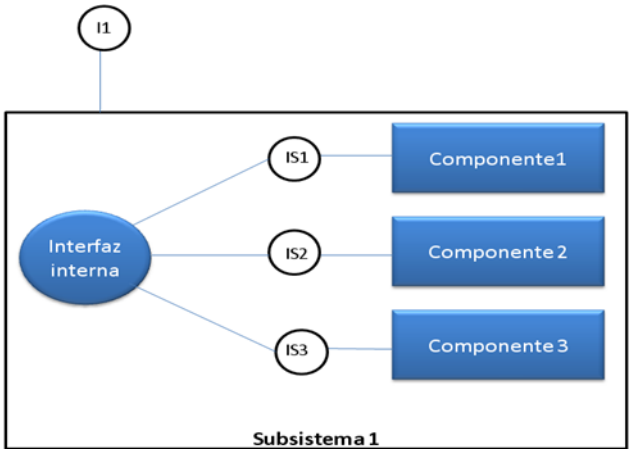


Figura 12 Integración entre componentes de un subsistema

Cada componente que brinde servicios tendrá una interfaz, la cual está constituida por una o varias clases donde se definen todos los servicios que brinda el componente. La nomenclatura de estas clases es <Nombre del concepto>service, por ejemplo EstructuraService, donde se van a ubicar los servicios relacionados con las estructuras. Por cada servicio se debe referenciar algún método de una clase model o un método de alguna clase domain en caso que lo único que se quiera es hacer una operación directa en la base de datos, esta política ayudará a evitar desarrollos posteriores, cumplir con la responsabilidad de las clases, entre otras ventajas. En resumen, nunca se debe implementar el flujo de un servicio en esta clase.

En el caso de la solicitud de servicios, en la actualidad se realiza desde la clase que lo necesita. A partir de ahora eso cambiará y se incluirá dentro de la interfaz del componente una clase para que sea la que se encargue de buscar los servicios, esta clase se nombrará ServiceOut**.** Por cada servicio que necesita el componente se creará un método que es el que se encargará de comunicarse con el IOC o el IOC Interno y devolver los datos que necesita el componente. Como conclusión cuando se requiera de un servicio en alguna clase model o controler se la solicitará a la clase serviceOut**.**

La separación de las clases services del resto permite que en las services sólo trabaje el arquitecto de componentes del proyecto, quien es el mayor responsable de la integración. Los programadores así se pueden abstraer del proceso de integración y concentrarse en sus tareas solamente. Otro beneficio está relacionado con los cambios,

pues si cambia algún servicio que un componente consume es más fácil localizar donde tiene que cambiar, o si se decide que cambie el origen del servicio a consumir es más sencillo el cambio.

Para ayudar en la verificación de que se cumplan estas definiciones, se debe añadir una herramienta automatizada que inspeccione la adherencia a los estándares definidos. Dentro de las herramientas que se pudieran usar se encuentra JDepend o NDepend. Por otra parte se pudiera configurar en la herramienta para el control de versiones, que en este caso es el Subversion que sólo puedan subir cambios en las clases services y los ficheros IOC y el IOC interno de cada proyecto los arquitectos.

2.3. Lineamientos

Se deben cumplir varios lineamientos para reducir los riesgos provocados por el proceso de integración. El primero de ellos es que se debe actualizar y subir constantemente los cambios realizados en las copias locales por cada desarrollador varias veces al día. Esta política permitirá que se pueda detectar rápidamente si el nuevo código en un componente provoca errores en el resto del sistema y sea más fácil de resolver esos errores porque se detectaron rápidamente.

Otra regla importante es que cuando un componente brinda un servicio no debe cambiarlo, a no ser un caso excepcional y en ese caso debe asegurarse de informar a todos los arquitectos de componentes que ha cambiado el servicio.

Otro de los lineamientos es que los desarrolladores deben asegurarse de no subir código defectuoso al servidor. Relacionado con lo anterior se debe arreglar lo más rápido posible los defectos encontrados producto de la integración. Para reducir el surgimiento de errores cada desarrollador debe realizar pruebas locales en su estación de trabajo antes de subir el nuevo código, para ello debe actualizar su copia local con el código del servidor donde están los últimos cambios realizados por el resto de los desarrolladores y asegurarse que en su copia local todo funciona correctamente.

Para ayudar con estos elementos se debe incorporar al proyecto un servidor con una aplicación de integración continua, que se ejecute cada vez que se sube nuevo código al servidor de control de versiones. Cada vez que detecte errores debe notificarle por correo al desarrollador que subió el código que provocó el fallo en el sistema.

Se propone que se use con este fin la herramienta JMeter, de la cual ya se tienen algunos conocimientos en el proyecto y se ha aplicado eventualmente, se propone establecer de manera formal su uso. Estos lineamientos están inspirados por lo definido en las prácticas para la integración continua, la que se explicó en el epígrafe 1.4.2.1.1.

2.4. Roles

Para determinar los roles con mayor responsabilidad en la gestión y desarrollo de la integración se tuvo en cuenta los roles que ya estaban definidos en el centro. Se estimó pertinente no proponer nuevos roles que se dediquen sólo a la integración porque dificultaría la relación con los ya existentes y retrasarían las tareas de integración, pues sería complejo establecer una frontera en sus responsabilidades. Se definen como roles responsables de la integración: arquitecto de componentes, arquitecto de sistema y desarrolladores. A continuación se explican las responsabilidades de esos roles, enfatizando en su vinculación con el área de integración. Además de los roles descritos también es importante el papel de los roles que se encargan de controlar el trabajo, en ese caso están: jefe de proyecto, líder de gestión y jefe de línea. Como elemento a resaltar se tiene que estos roles deben poseer valores humanos tales como responsabilidad y compromiso con el equipo, estos valores contribuirán a que el todo el equipo sienta respeto por el trabajo de sus compañeros y comprenda que si no realiza su trabajo correctamente puede perjudicar al equipo y al éxito del proyecto. Visto de forma práctica la formación de estos valores ayudará a mejorar la calidad del proceso de desarrollo y del producto final.

2.4.1. Arquitecto de componentes

El arquitecto de componentes es el principal responsable de la integración dentro de un proyecto, desempeña las tareas que estaban definidas para un arquitecto de proyecto y ahora desempeñará las tareas propuestas para gestionar la integración de componentes, tanto interna como externa.

- Competencias:
  - Conocimiento avanzado de programación en php.
  - Conocimiento del negocio del proyecto y una visión general de los procesos fundamentales del proyecto.
  - Conocimiento de patrones de diseño y arquitectura.
  - Conocimiento de estilos arquitectónicos, fundamentalmente del MVC y del desarrollo por componentes.

- Amplio dominio de los mecanismos de integración de componentes (externa e interna).
- Dominio de las clases usadas para publicación de servicios, clases services, xml IOC.
- Buena capacidad de comunicación.
- Capacidad de discutir elementos técnicos y defender sus propuestas y aceptar la de otros.

❖ Responsabilidades:

- Participar en el diseño por componentes de la solución del proyecto.
- Participar en los talleres de levantamiento de servicios por cada componente que se desarrolle en el proyecto.
- Realizar las solicitudes de servicios a otros proyectos y atender los servicios solicitados.
- Realizar la solicitud de servicios externos al arquitecto de sistema.
- Participar en las actividades para coordinar los detalles de los servicios solicitados y los que solicite.
- Reportar las no conformidades que se detecten en el proyecto y resolver las que se le reporten.
- Asignar tareas de desarrollo a los arquitectos, tanto de nuevos desarrollos como de solución de problemas en alguna función implementada, además debe darle seguimiento a esas tareas.
- Actualizar las clases Services, ServiceOut, el IOC externo e interno con los servicios de los componentes del proyecto.
- Informar a la dirección del proyecto e insertar desviaciones cuando existan atrasos en el proyecto por causa de la integración.
- Llenar y actualizar los artefactos correspondientes.

2.4.2. Arquitecto de sistema

El arquitecto de sistema es el rol que tiene una vista global de todo el sistema y de los componentes que lo conforman. Es el coordinador de la integración con otros sistemas. Es el encargado de velar porque se sigan los

estándares especificados para el desarrollo y controlar el trabajo de los arquitectos de componentes en los proyectos.

- Competencias
  - Conocimiento avanzado de patrones de diseño y arquitectura.
  - Conocimiento avanzado de estilos arquitectónicos, fundamentalmente del MVC y del desarrollo por componentes.
  - Amplio dominio de los mecanismos de integración de componentes (externa e interna).
  - Buena capacidad de comunicación.
  - Conocimiento de los procesos fundamentales de todos los proyectos.
  - Visión de otros sistemas, que puedan tener integración con el suyo.
- Responsabilidades
  - Controlar el trabajo de los arquitectos de componente.
  - Asignar tareas a los arquitectos de componente y darle seguimiento a esas tareas.
  - Participar en el diseño por componentes de los proyectos.
  - Participar en los talleres para revisar cuando un servicio tiene implicaciones en el sistema y hay necesidad de nuevos desarrollos.
  - Atender las solicitudes de servicios externos por los arquitectos de componente.
  - Atender las solicitudes de servicios que le realizan otros sistemas.
  - Solicitar servicios a sistemas externos.
  - Participar en los talleres con los arquitectos de componente para precisar detalles de los servicios externos solicitados.

- Participar en los talleres con personal de otros sistemas para precisar detalles de servicios que le son solicitados y los que él solicita.
- Definir los mecanismos para utilizar servicios externos y explicarle a los arquitectos de proyecto.
- Llenar y actualizar los artefactos correspondientes.

2.4.3. Desarrollador

El desarrollador implementará las funciones que le asigna el arquitecto de componentes de su proyecto. Aunque no es el rol de más responsabilidad, su trabajo es determinante para la calidad del proceso de integración.

❖ Competencias

- Conocimiento de php.
- Dominio de la estructura de los componentes.
- Conocimiento de los procesos de negocio asociados a los componentes donde trabaja.
- Dominio de los estándares de desarrollo del proyecto.

❖ Responsabilidades

- Desarrollar y probar las funciones que le asigne el arquitecto de componentes de su proyecto.
- Notificar al arquitecto de componentes del proyecto cuando detecte la necesidad de nuevos servicios.
- Notificar al arquitecto de componentes del proyecto cuando detecte errores en alguno de los servicios que utiliza su componente.
- Resolver las no conformidades en las funciones que se utilizan en los servicios que el arquitecto de componentes le asigne.
- Cumplir con los estándares de desarrollo del proyecto.

2.5. Actividades

Las actividades propuestas están encaminadas a que se siga un flujo formal y coherente para efectuar la integración de componentes. Se definen tareas encaminadas a gestionar la integración entre proyectos, la integración interna y la integración con otros sistemas. El cumplimiento de este flujo de actividades contribuirá a mejorar el proceso de integración de componentes y por consiguiente ayudará a mejorar el proceso de desarrollo en general. Un elemento que se tuvo en cuenta para diseñar este flujo de actividades es uno de los principios del desarrollo ágil del software presentado en el epígrafe 1.4 y que expresa que el método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

2.5.1. Taller de levantamiento de servicios

En este taller participará el analista del proyecto, el arquitecto de componentes u otro rol que tenga conocimiento de los procesos de cada componente que integra el proyecto. El taller se realizará luego de haber efectuado el diseño por componentes del proyecto, el cual debe haber sido revisado por el arquitecto de sistema. En este taller con los conocimientos de las personas se realizará un levantamiento de todos los servicios que debería brindar cada componente del sistema y los que debe consumir. Se debe detallar lo más que se pueda para lograr que se identifique la mayor cantidad de servicios. Esta actividad es significativa para el proceso de integración porque es más fácil gestionar los servicios que son identificados desde el principio. Un resultado ideal de esta actividad es que se determine el cien por ciento de los servicios necesarios por componente al igual que los que debe desarrollar. Como salida de esta actividad se llenará los artefactos Servicios consumidos por componente y Servicios brindados por componente.

2.5.2. Revisión del estado de servicios

Esta actividad se ejecutará cada vez que se identifique la necesidad de un nuevo servicio en un proyecto, ya sea que se haya identificado el nuevo servicio en el taller de levantamiento de servicios o durante el desarrollo. Para facilitar la búsqueda del estado de un servicio se utilizará el módulo para la gestión de servicios, donde existen varias opciones de búsqueda que permiten verificar si el servicio que se necesita está implementado.

La importancia de esta tarea radica en que cuando se necesite un servicio, primero se debe verificar si ya está implementado y no solicitar que se implemente un nuevo servicio, que es lo que se hace instintivamente. Con esta

actividad se logra que se haga una solicitud de servicio sólo cuando se esté seguro que no está implementado. Como salida se actualizará el artefacto Servicios por componente**.**

2.5.3. Probar y evaluar servicios

Esta actividad se ejecutará cuando se revise el estado de un servicio y se verifique que ya está implementado o cuando se notifica que un servicio solicitado ya está disponible. Para probar se utilizará el módulo para la gestión de servicios. Luego se actualizará el artefacto Servicios por componente con el resultado de la prueba**.**

Con la realización de esta actividad se reducirá la posibilidad de usar un servicio que tenga errores, ya sea de código o de negocio.

2.5.4. Asignar no conformidad de integración

Esta actividad se realiza cuando se detecta alguna no conformidad en un servicio, puede ser detectada mediante la revisión usando el módulo para la gestión de servicios o que se detecte un error en algún servicio que ya se estaba utilizando. Se asignará mediante el Redmine una no conformidad de integración al arquitecto de componentes del proyecto que brinda el servicio.

Entre más rápido se asigne la no conformidad de integración más rápido se debe resolver porque menos tiempo ha pasado desde que se introdujo el error, facilitando la retroalimentación. Además entre más rápido se resuelva menos componentes afectará el error. El arquitecto de componentes del proyecto es el encargado de la actividad.

2.5.5. Comunicar tiempo de solución de no conformidad

Cuando un arquitecto de componentes recibe la notificación de una no conformidad en uno de los servicios que brinda su proyecto debe evaluar el problema y comunicar al arquitecto que reportó la no conformidad el tiempo que demorará resolver el problema. Esta tarea tiene como relevancia que permite la comunicación entre los integrantes del proyecto, la planificación del tiempo de solución de la no conformidad la determina el arquitecto de componentes con los involucrados en sus proyectos y no se le planifica un tiempo por otra persona que no tiene conocimientos de los detalles de la implementación.

2.5.6. Asignar tarea de resolver no conformidad en una función que utiliza algún servicio

Cuando exista alguna no conformidad en alguno de los servicios el arquitecto de componentes del proyecto acordará con el desarrollador que la resolverá el tiempo de solución y le asignará la tarea mediante el Redmine, para ello creará una tarea de integración, además la relacionará con la no conformidad que la generó. Lo más recomendable es asignarle la tarea al desarrollador que implementó la función que utiliza el servicio.

2.5.7. Solicitud de servicio

Después que el arquitecto de componentes de un proyecto comprueba que alguno de los servicios que necesita no está implementado, le asignará mediante el Redmine una tarea de integración al arquitecto de componentes del proyecto que debe brindarlo. Aprovechando la rapidez de la comunicación mediante el Redmine la solicitud se atenderá rápidamente.

2.5.8. Encuentro para solicitud de servicios

Después que se haya realizado una solicitud de un servicio, los arquitectos de componentes de los dos proyectos coordinarán un encuentro para definir los detalles del servicio. En ese encuentro se revisarán todos los detalles del servicio y el tiempo en implementarse. Como salida de la actividad se llenará el artefacto solicitud de servicios.

Esta es otra de las tareas que promueven la comunicación entre los integrantes del proyecto para planificar el tiempo de las tareas y no fijar de antemano un tiempo que después no pueda ser cumplido. Señalar además que de la calidad de esta actividad dependerá en gran medida la calidad del servicio a implementar.

2.5.9. Revisión de implicaciones de la implementación de un servicio

Después de efectuar la reunión para la solicitud de un servicio, el arquitecto de componentes del proyecto que brindará el servicio debe revisar si el nuevo servicio implica algún cambio en el diseño o implementación de alguno de los componentes de su proyecto.

Esta tarea tiene gran impacto en el buen diseño de los subsistemas y reconoce que el cambio es inminente, por lo que hay que planificar tareas para que sea menos traumático y lo más organizado posible.

2.5.10. Taller para revisar implicaciones de la implementación de un servicio

Si el arquitecto de componentes determina que la implementación de un servicio provoca algún cambio en el diseño de los componentes, se efectuará un taller con el arquitecto de sistema. En este taller se determinará los cambios necesarios en el diseño de los componentes y se documentará el origen de los cambios, además se generarán tareas de implementación para implementarlos.

2.5.11. Revisión de la existencia de una función que resuelva las necesidades de un servicio

Cuando es necesario implementar un nuevo servicio, el arquitecto de componentes debe revisar si alguna de las funciones que ya existen le da solución a las necesidades del servicio. El planteamiento explícito de esta tarea ayudará a que no se duplique el código y que no existan funciones que realicen lo mismo. Si no se define como lo planteado lo más probable es que no se revise.

2.5.12. Referenciar una función en la clase service del componente y el servicio en el xml IOC

Cuando la función que resuelve las necesidades de un servicio está implementada, ya sea porque estaba de antemano o porque se le asignó la tarea a un desarrollador cuando llegó la solicitud del servicio, el arquitecto de componentes referenciará la función en la clase services del componente. Luego incluirá el servicio en el xml IOC, para ello utilizará el módulo para la gestión de servicios.

El cumplimiento de esta tarea permite que sólo los arquitectos de componente escriban en las clases services de los componentes y en el xml IOC, lo que reduce el riesgo de que se introduzcan errores en esos ficheros, además que el arquitecto de componentes lleve el control de todos los servicios que brindan los componentes de su proyecto. Por otra parte la utilización del módulo para la gestión de servicios para que inserte en el xml IOC también reduce las posibilidades de inserción de errores en este fichero. Sólo en momentos excepcionales un desarrollador ejecutará esta tarea con la autorización del arquitecto de componentes y el jefe de proyecto.

2.5.13. Referenciar una función en la clase service del componente y el servicio en el xml IOC interno

Cuando la función que resuelve las necesidades de un servicio interno está implementada, ya sea porque estaba de antemano o porque se le asignó la tarea a un desarrollador cuando llegó la solicitud del servicio, el arquitecto de

componentes referenciará la función en la clase services del componente. Luego incluirá el servicio en el xml IOC interno del proyecto, para ello utilizará el módulo para la gestión de servicios.

El cumplimiento de esta tarea permite que sólo los arquitectos de componente escriban en las clases services de los componentes y en el xml IOC interno, lo que reduce el riesgo de que se introduzcan errores en esos ficheros, además que el arquitecto de componentes lleve el control de todos los servicios que brindan los componentes de su proyecto. Sólo en momentos excepcionales un desarrollador ejecutará esta tarea con la autorización del arquitecto de componentes y el jefe de proyecto.

2.5.14. Asignar tarea de desarrollo de una función para resolver necesidades de un servicio

Cuando un arquitecto compruebe que no existe ninguna función implementada que satisfaga las necesidades de un servicio solicitado, le asignará la tarea de implementar esta función a un desarrollador. La asignación de la tarea se realizará mediante el Redmine, donde además se le especificarán los detalles de la función y el tiempo para su cumplimiento. Cuando se cree la tarea se debe relacionar con la solicitud del servicio que realizó el arquitecto de componentes del proyecto que lo necesita, esta relación le permitirá al arquitecto de componentes que solicitó el servicio conocer el estado de la implementación del servicio requerido y en dependencia de ella tomar algunas decisiones, como puede ser insertar riesgos de integración si existe alguna anormalidad.

Es importante que la tarea se le asigne a uno de los desarrolladores que trabaja en el componente que brindará el servicio porque eso reducirá el tiempo de desarrollo de la función y reducirá el riesgo de introducir errores, todo eso justificado por el hecho de que los desarrolladores del componente son lo que conocen lo detalles de su implementación.

2.5.15. Desarrollar y probar función para resolver necesidades de un servicio

Cuando se le asigna la tarea de implementar una función para un servicio a un desarrollador, este debe revisar los detalles de la función. Cuando haya terminado la implementación, debe probar lo realizado y sólo cuando esté seguro de que funciona correctamente cerrará la tarea en el Redmine.

Además de la implementación en sí, es muy importante la prueba, pues de hacerse correctamente evitará errores en el futuro que se traduce en tiempo y costo para el proyecto.

2.5.16. Notificación de la disponibilidad de un servicio

Cuando el arquitecto que brinda el servicio halla incluido el nuevo servicio en la xml IOC, mediante el módulo de gestión de servicios y se asegure que el servicio funciona correctamente apoyándose en la misma herramienta, cerrará la tarea donde se solicitaba el nuevo servicio en el Redmine.

Como elemento a destacar en esta tarea está el hecho de que la notificación llegará de inmediato al arquitecto de componentes que solicitó el servicio, demostrando la efectividad del mecanismo de comunicación. Además señalar el hecho de la prueba del servicio por parte del arquitecto de componentes que lo brinda, lo que asegura que el servicio funciona correctamente. Como resultado de esta actividad se actualizará también el artefacto de servicios brindado por componente.

2.5.17. Probar y evaluar un servicio

Cuando se notifica a un arquitecto de componentes que el servicio solicitado ya está disponible o cuando él detecta que un servicio que necesita ya está implementado, probará el servicio apoyándose en el módulo para la gestión de servicios. En esta prueba se debe asegurar no sólo que el servicio funciona sino que además funciona correctamente desde el punto de vista de negocio.

Con esta actividad se garantiza que no se use ningún servicio que tenga problemas, tanto de implementación como funcionales. Además ayuda a mejorar la retroalimentación. Como salida de esta actividad se actualizará el artefacto servicios por componente en dependencia del resultado de la prueba al servicio. Si el resultado de la prueba es positivo se le informará a los desarrolladores que utilizarán el servicio.

Esta tarea también se desarrollará cuando se le notifica a un arquitecto que alguno de los servicios que utiliza ha sido cambiado.

2.5.18. Asignar Implementación de cambio en función usada en servicio

Uno de las reglas que posee el desarrollo por componentes, es que no se debe cambiar un servicio que brinda un componente. No siempre se puede cumplir esta regla, sobre todo si el componente que brinda el servicio está en desarrollo. Se debe hacer todo lo posible porque se cumpla la regla, pero en caso que sea inminente el cambio, se debe realizar organizadamente para que el impacto en el sistema sea el menor posible.

Después de haber identificado los elementos a cambiar el arquitecto de componentes del proyecto le asignará mediante el Redmine la tarea de tipo de integración de cambiar la función que brinda el servicio al desarrollador que lo implementó. Aquí debe especificarle los detalles que hay que cambiar y el tiempo para su solución.

2.5.19. Implementación de cambio en función usada en servicio

Cuando se le asigne una tarea de implementar un cambio a un desarrollador debe realizar los cambios, probar la función nuevamente y cerrar la tarea en el Redmine.

2.5.20. Notificación de cambio en un servicio

Cuando el desarrollador notifique que ha implementado el cambio, el arquitecto de componentes probará nuevamente el servicio mediante el módulo de gestión de servicios para asegurarse que el servicio funciona correctamente, luego insertará una tarea de integración a todos los arquitectos de componente explicando que el servicio ha cambiado, los motivos del cambio y la nueva forma de usarlo. Se actualizará además el artefacto servicios por componente con las nuevas especificaciones del servicio.

2.5.21. Insertar desviaciones de integración

En el caso que se necesite un servicio en el desarrollo de un componente y el tiempo de desarrollo del servicio se prolongue más de lo que se había acordado, el arquitecto de componentes insertará una desviación de tipo integración en el Redmine. Lo mismo se hará en caso de que se detecten no conformidades en los servicios que se utilizan y sus soluciones se prolonguen más de un día.

También se insertarán desviaciones de integración cuando sea necesario priorizar las tareas de integración y dejar el resto de las tareas de desarrollo para más adelante.

2.5.22. Solicitud de servicio externo

Cuando se identifique la necesidad de un servicio de otro sistema, ya sea en el taller de levantamiento de servicios o posteriormente, el arquitecto de componentes del proyecto se lo solicitará al arquitecto de sistema. Para ello creará una tarea del tipo integración en el Redmine donde especificará los detalles del servicio que necesita.

El tiempo en desarrollar el servicio externo es un poco más complicado porque dependerá del trabajo en otro sistema.

2.5.23. Encuentro con arquitecto de sistema externo para solicitud de servicio

Cuando al arquitecto de sistema uno de los arquitectos de componente le solicita un servicio externo debe efectuar un encuentro con alguna persona encargada de ese tema en el proyecto que debería brindar el servicio. Se pondrán de acuerdo en cuanto a los detalles del servicio y acordarán la fecha en que estará disponible el servicio.

El arquitecto de sistema actualizará la tarea de solicitud de servicio poniéndole como fecha de fin la acordada. Se llenará el artefacto Solicitud de servicios externos.

2.5.24. Notificación de la disponibilidad de un servicio externo

Cuando al arquitecto de sistema se le notifique que alguno de los servicios solicitados ya está disponible, lo probará. Si existe alguna no conformidad se la enviará al arquitecto del sistema que brinda el servicio. De estar correcto el servicio, notificará al arquitecto de componentes que realizó la solicitud, para ello sólo debe cerrar la tarea en el Redmine donde se solicitaba el servicio. Se actualizará el artefacto servicios externos.

2.5.25. Detección de necesidad de nuevo servicio

Durante el desarrollo de un componente se puede detectar la necesidad de nuevos servicios que no fueron identificados durante el taller de levantamiento de servicios. Generalmente la necesidad de nuevos servicios la detectará el desarrollador que implementa el componente, en ese caso se lo notificará al arquitecto de componentes para que gestione el servicio.

2.6. Artefactos

2.6.1. Servicios consumidos por componente

En el artefacto servicios consumidos por componente se recogerán los datos de todos los servicios que utiliza un componente determinado. Entre los datos que se guardan por cada servicio está el nombre, la descripción, los parámetros de entrada, el resultado, el origen del servicio y el estado en que se encuentra el servicio.

Uno de los elementos a resaltar de este artefacto es que ayudará a la portabilidad del componente, porque si es necesario utilizarlo en otro contexto sería suficiente con leer el artefacto para conocer las dependencias del componente y no habría que revisar el código. Este artefacto contribuye a que los componentes estén correctamente documentados, lo que constituye uno de los lineamientos especificados en el epígrafe 1.2.2. Además, si existe algún cambio en un servicio es más fácil identificar si el componente lo utiliza. Los detalles del artefacto se encuentran en el paquete (Entregables 2010) en la sección referente a los artefactos.

2.6.2. Servicios brindados por componente

En el artefacto servicios brindados por componente se recogerán los datos de todos los servicios que brinda determinado componente. Entre los datos que se recogen en el artefacto está el nombre del servicio, los parámetros de entrada, el resultado y el estado en que se encuentra el servicio en ese momento.

Igualmente este es un artefacto de gran impacto para la reutilización y portabilidad de un componente porque mediante su consulta se tiene los datos necesarios referentes a los servicios que brinda el componente y no sería necesaria revisar en la interfaz del componente. Como es un documento permite que cualquier persona interesada, aunque no tenga conocimientos informáticos pueda consultarlo y verificar si alguno de los servicios le interesa. Los detalles del artefacto se encuentran en el paquete (Entregables 2010) en la sección referente a los artefactos.

2.6.3. Solicitud de servicios

El artefacto Solicitud de servicios es una especie de contrato, donde se van a definir los términos de ambas partes, del solicitante y el que brindará el servicio. Se recogerán algunos datos como la fecha de solicitud, nombre del servicio, parámetros de entrada, resultado, fecha en que estará disponible y los datos de las personas que participan en el encuentro.

Este artefacto ayudará a evitar problemas en el futuro, pues quedará la constancia de lo que se convenio entre las dos partes y contribuirá a la buena gestión de la gestión de la integración. Los detalles del artefacto se encuentran en el paquete (Entregables 2010) en la sección referente a los artefactos.

2.6.4. Servicios externos consumidos

En el artefacto servicios externos consumidos se recoge los datos de todos los servicios que usa el sistema de otros sistemas. Entre los datos que se recogen está el nombre del servicio, descripción, parámetros de entrada, resultado, estado, sistema que los brinda.

La mayor importancia del artefacto es que se tiene en un mismo lugar todos los servicios externos que consume el sistema, lo que ayuda a revisar las dependencias que tiene el sistema, además se tienen detalles de los servicios lo que facilita el proceso si es necesario algún cambio. Los detalles del artefacto se encuentran en el paquete (Entregables 2010) en la sección referente a los artefactos.

2.6.5. Servicios externos brindados

En el artefacto servicios externos brindados se recogen los datos de todos los servicios que brinda el sistema a otros sistemas. Entre los datos que se recogen está el nombre del servicio, descripción, parámetros de entrada, resultado, estado.

Este artefacto constituye una guía para revisar todos los servicios que brinda el sistema para el exterior. Puede ser de utilidad para otros sistemas que deseen consumir algún servicio de los que se brinda. Los detalles del artefacto se encuentran en el paquete (Entregables 2010) en la sección referente a los artefactos.

2.7. Mecanismos para control de calidad del proceso

En el libro de procesos realizado sobre el área de procesos de integración se definen métricas para comprobar cuantitativamente cuál es el porciento de cumplimiento de lo definido en el proceso. Además se define una lista de chequeo que de forma rápida y sencilla permite evaluar como se desarrolla el proceso en un proyecto. A continuación se presenta la lista de chequeo.

- Está definido el arquitecto por cada proyecto y el arquitecto de sistema central.
- Están actualizados todos los artefactos relacionados con la integración.
- Se cumplen con los estándares definidos.
- Las clases services sólo son actualizadas por el arquitecto del proyecto.

- Está actualizado el Redmine con todas las tareas de integración.
- El arquitecto de componentes del proyecto tiene tareas fundamentalmente de integración.

2.7.1. Herramientas de Apoyo

En el epígrafe 1.5 se analizó la importancia que tenía la automatización del proceso de desarrollo para mejorar su calidad y eficiencia. En este sentido se proponen algunas herramientas que pueden contribuir a mejorar el proceso de integración. También tienen un alto impacto en la toma de decisiones porque se definen indicadores que se monitorearán de forma automática y se podrá detectar irregularidades en el proceso, permitiendo que las personas tengan elementos para ejecutar ciertas acciones.

2.7.1.1. Redmine

Para que el Redmine apoyara la gestión de la integración fue necesario realizarle algunas configuraciones con el fin de configurar los datos relevantes para la ejecución y control del proceso. Fue necesario incluir un nuevo tipo de tarea, nombrado tarea de "integración". Este tipo de tarea se le asignará a todas las actividades relacionadas con la integración que se realizan en los proyectos. Se incluyó un nuevo tema de tarea, denominado "no conformidad de integración". Igualmente se incluyó un nuevo tipo de desviación nombrada "integración".

Se le debe incluir al Redmine la capacidad de enviar alertas sobre irregularidades en el proceso de integración. Las reglas definidas para las alertas son las siguientes:

- Cuando se haga la planificación de un proyecto y no existan tareas del tipo integración.
- Cuando se haga solicitud de un servicio y en el proyecto al que se solicitó le asignen la tarea a un desarrollador que no tenga fondo de tiempo para implementarlo.
- Cuando se inserta una no conformidad a un servicio y no es actualizada en el día se debe avisar al arquitecto del proyecto solicitante y al que brinda el servicio.
- Cuando se le asigne la tarea a un desarrollador de la implementación de un servicio y el día de finalización no se cierre la tarea, se debe avisar al arquitecto del proyecto que desarrolla el servicio y al que lo solicitó. Se debe

además insertar una desviación del tipo integración si por este motivo se producen atrasos en el proyecto que consume el servicio.

- Se debe alertar al jefe de proyecto, de línea y arquitecto de sistema si en un proyecto no existe nadie con el rol de arquitecto de componente. También se debe generar automáticamente un riesgo en el Redmine por esta causa.
- Cuando al arquitecto de un proyecto se le planifica todo el tiempo en una semana y no se le deja tiempo para las tareas de integración que surgen. También se debe generar automáticamente un riesgo en el Redmine por esta causa y enviar una alerta al jefe del proyecto y al arquitecto de sistema.
- Cuando se le asignan muchas tareas que no son de integración al arquitecto de algún proyecto se le debe alertar al arquitecto de sistema, para que revise esta situación. También se debe generar automáticamente un riesgo en el Redmine por esta causa. Considerando que el tiempo promedio para cumplir una tarea es de 2 horas y que el fondo de tiempo total es de 44 horas, se considera que en una semana el arquitecto no debe tener más de 10 de tareas que no sean de integración.
- Cuando se asigne una tarea de integración a una persona que no tenga las competencias para realizarla, se debe notificar al arquitecto de componentes del proyecto que lo solicitó y al del proyecto que lo brinda. También se debe generar automáticamente un riesgo en el Redmine por esta causa. Para poder cumplir con esta regla al Redmine se le debe adicionar la funcionalidad de asignar competencias a las personas y además poder definir competencias básicas para un rol. Con estos elementos se podría verificar si una persona tiene las competencias para desarrollar una tarea, en este caso específico de integración.
- Cuando un componente necesite servicios, la fecha de finalización del componente no puede ser menor que la fecha de finalización de los servicios que necesita. En caso que a alguno de los servicios se le ponga fecha de finalización superior a la de finalización del componente se debe alertar al arquitecto de componentes del proyecto que solicita el servicio.
- Si una persona tiene tareas de integración asignadas a cumplir en un período de tiempo y por algún motivo se va a ausentar parte de ese tiempo, se debe crear automáticamente un riesgo de integración en todos los proyectos que esperan servicios con los que está relacionada esa persona, además alertarle a los arquitectos de componente de esos proyectos. Para lograr esto se debe adicionar al Redmine la posibilidad de registrar

ausencias futuras, es decir que se pueda registrar que una persona va a estar ausente durante un período de tiempo determinado.

- Si se le asigna una tarea de integración a una persona que va estar ausente durante el período que se definió que se implementaría el servicio, se alertará al arquitecto de componentes que solicitó el servicio y se creará un riesgo de integración.

2.7.1.2. Módulo de gestión de servicios

El módulo de gestión de servicios se desarrolló con el objetivo de ayudar en el proceso de integración durante el desarrollo. Las funcionalidades están encaminadas a que los desarrolladores cuenten con una herramienta que les facilite el trabajo, reduzca los errores introducidos en la codificación y que contribuya a que no se duplique el código. Para lograr estos objetivos el módulo posee las siguientes funcionalidades:

- Búsqueda avanzada de servicios.
- Prueba de servicios.
- Publicar nuevos servicios.
- Modificar servicios publicados.
- Reportar no conformidades.

Como elemento a destacar en esta herramienta se encuentra la posibilidad de integración con el Redmine. Para lograrlo se utiliza la interfaz de servicios del Redmine que permite la creación de peticiones por esa vía. El sistema permite que cuando se pruebe un servicio y tenga errores, ya sea que no funciona o que devuelve un resultado incorrecto se le puede asignar una no conformidad de integración al arquitecto de componentes del proyecto que brinda el servicio, en caso que sea un servicio interno se le asigna al arquitecto del propio proyecto el cual se encargará de gestionar su solución, igualmente se puede insertar un desviación por integración si se detectan fallos en un servicio y genera un atraso en el proyecto. En el caso que se modifique un servicio, también se puede generar una tarea de integración para todos los arquitectos de componentes informando el servicio que ha cambiado y para que revisen si en su proyecto se usa. La descripción de los requisitos que cumple esta aplicación se puede encontrar en el paquete (Entregables 2010) en la sección referente al módulo de gestión de servicios.

2.7.1.3. Módulo para la publicación de servicios externos

El módulo de gestión de servicios externos se desarrolló con el objetivo de ayudar en el proceso de integración con sistemas externos. Con este módulo se abstrae a los desarrolladores de algunas tareas, incluyendo la generación de código. Con este fin se implementaron las siguientes funcionalidades:

- Publicación de servicios externos.
- Prueba de servicios externos.
- Generación de paquetes de servicios.

La descripción de los requisitos que cumple esta aplicación se puede encontrar en el paquete (Entregables 2010) en la sección referente al módulo de gestión de servicios externos.

2.8. Conclusiones del capítulo

- A partir de los elementos teóricos estudiados se elaboró una guía metodológica encaminada a gestionar la integración de componentes en los proyectos del sistema CEDRUX, la aplicación de esta guía contribuirá a mejorar la calidad del proceso de desarrollo en estos proyectos. Dentro de los elementos más importantes incluidos en la guía se encuentran: el flujo de actividades para gestionar la integración, los roles responsables de la integración, los artefactos resultantes del proceso, los lineamientos y estándares que regirán el proceso y algunas herramientas propuestas de apoyo para facilitar el proceso.
- Para el desarrollo de la guía se tuvo en cuenta las características particulares del proyecto y de los elementos recogidos en las entrevistas realizadas, lo que demuestra que la guía tiene un alto nivel práctico.

Capítulo 3 Aplicación de la guía

3.1 Introducción

En este capítulo se describen los resultados de la ejecución del experimento realizado para comprobar el impacto de la aplicación de la propuesta en la eficiencia del proceso de desarrollo en los proyectos. Además se muestra los resultados de la evaluación de la guía teniendo en cuenta los indicadores definidos.

3.2 Ejecución del experimento

Se realizó un experimentó para determinar el efecto causado por la aplicación de la guía metodológica propuesta en la eficiencia del proceso de desarrollo de software en los proyectos. El experimento se desarrolló en condiciones naturales. Para determinar la eficiencia del proceso de desarrollo se tuvieron en cuenta los siguientes indicadores:

- Tiempo promedio de implementación de servicios: Significa el tiempo promedio de implementación de un servicio, medido en días.
- Tiempo promedio de gestión de servicios: Se refiere al tiempo transcurrido desde que se detecta la necesidad de un nuevo servicio hasta que se notifica la disponibilidad del mismo.
- Índice de desviaciones: Es la razón entre la cantidad de desviaciones por integración en un proyecto sobre la cantidad de servicios consumidos.
- Índice de no conformidades por servicios brindados: Es la razón entre el número de no conformidades de integración en los servicios brindados y el total de servicios brindados.
- Índice de no conformidades por servicios consumidos: Es la razón entre el número de no conformidades de integración en los servicios consumidos y el total de servicios consumidos.
- Tiempo promedio de gestión de no conformidades de integración: Se refiere al tiempo que transcurre desde que se detecta una no conformidad de integración hasta que se notifica su solución.

La población total está constituida por todos los proyectos del sistema CEDRUX, para un total de diez (10). La muestra se seleccionó de forma intencionada teniendo en cuenta que todos los elementos de la población tienen

características muy similares, además se seleccionaron los proyectos en los cuales se pudieran obtener los datos necesarios en el período antes de aplicar la propuesta. La muestra quedó constituida por siete (7) proyectos, lo que representa un setenta por ciento de la población total.

Se recogieron los datos durante cuarenta días después de introducir progresivamente la guía propuesta y se recogió el comportamiento de los indicadores definidos. Para obtener los datos se diseñó un instrumento de captación de datos y se aplicó a los arquitectos de los proyectos antes de comenzar el experimento y después de haber aplicado la guía, el instrumento aplicado se en encuentra en el Anexo 4.

3.2.1. Aplicación de la guía

Para la aplicación de la guía lo primero que se realizó fue capacitar al personal más involucrado en el proceso de integración. En este caso se hace referencia a los arquitectos de componentes de cada proyecto, arquitecto de sistema, jefe de proyectos y líderes de gestión de cada proyecto.

Para la capacitación se realizaron encuentros por cada proyecto, donde se explicó en qué consistía la propuesta y cómo se llevaría el proceso de integración a partir de ese momento, se les entregó la documentación donde se describe la propuesta detalladamente. También se realizaron ejercicios para mostrarle como se introducen los datos relacionados con la integración en el Redmine, además de hacer presentaciones del funcionamiento del módulo de gestión de servicios y se les entregó el manual de usuario. Cuando se culminó con la capacitación se procedió a la aplicación de la propuesta.

Un elemento a destacar fue que se integró el procedimiento a los procesos de gestión de la calidad del centro, para ello se elaboró el libro de procesos del área de procesos de integración, que constituye una de las áreas de procesos definidas en el nivel 3 de CMMI (Capability Maturity Model Integration) y para estar alineado al programa de mejoras que se sigue en el centro. Como parte de este libro se especifican los mecanismos para que la subdirección de calidad compruebe el grado de implementación de los procesos en las revisiones que se le realizan a los proyectos. Dentro de estos mecanismos se encuentran una lista de chequeo y una métrica que retorna el índice de implementación del proceso o adherencia al proceso en un proyecto. Además, la revisión de los artefactos definidos servirá para conocer la correspondencia entre la implementación de tareas de integración, la actualización del

Redmine y los artefactos resultantes, algo que no se podía conocer anteriormente. Los detalles del libro del área de procesos de integración se encuentran en el paquete (Entregables 2010) en la sección referente al libro de procesos.

3.2.2. Análisis de los resultados

Para demostrar el impacto de la aplicación de la guía, a continuación se realiza un análisis del comportamiento en cada uno de los indicadores determinados.

En el caso del indicador tiempo promedio de implementación de servicios, se muestra el resultado en la Tabla 1. En todos los casos, menos uno se redujo el tiempo de implementación, lo que demuestra que con la introducción de la propuesta se puede reducir este tiempo, lo que contribuiría a reducir el tiempo total de desarrollo en un proyecto, traduciéndose en reducción de los costos. En el caso del proyecto donde el tiempo en el segundo período es cero es porque en ese período no implementó ningún servicio nuevo, no obstante en el proyecto consideran que mediante los elementos planteados en la propuesta se debe reducir el tiempo de implementación. Un elemento a tener en cuenta es la desviación estándar, en el primer período es alta, lo que evidencia la variedad del proceso en cada uno de los proyectos, influenciado por sus particularidades. En el segundo período se reduce, lo que demuestra que existe la tendencia a homogeneizar este elemento en los proyectos.

Proyecto	Tiempo de desarrollo antes	Tiempo de desarrollo después
Activos Fijos Tangibles	1	0
Capital Humano	7	1
Estructura y Composición	7	3
Finanzas	3	1
Logística	3	1
Configuración	2	1
Costos y Proceso	4	1
Media	3,85714286	1,14285714
Varianza	5,47619048	0,80952381
Desviación estándar	2,1602469	0,81649658

Tabla 1 Tiempo promedio de implementación de servicios

En la Figura 13 se ilustran estos datos mediante una gráfica que ayuda a visualizar mejor la diferencia entre ambos períodos.

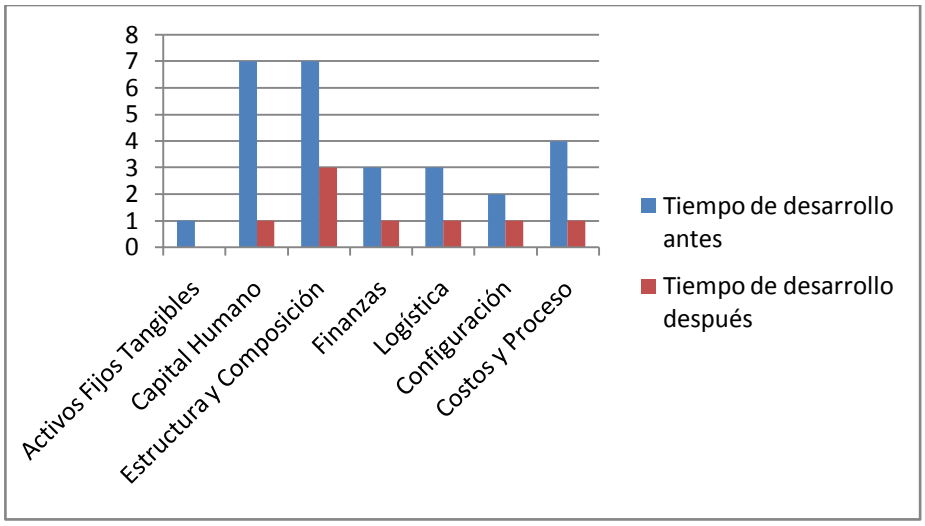


Figura 13 Tiempo promedio de implementación de servicios

El indicador tiempo de gestión de servicios describe varios elementos del proceso, tiene en cuenta desde que se detecta la necesidad de un nuevo servicio hasta que se informa que el servicio está disponible, permite comprobar las vías de comunicación entre los proyectos, la importancia de la integración, entre otros elementos. Los resultados se muestran en la Tabla 2. En cuatro de los proyectos se redujo el tiempo promedio de gestión de servicios, en algunos casos se redujo hasta más de un 50 % lo que evidencia el impacto positivo de la aplicación de la propuesta. En los otros 3 proyectos no se pudo comprobar porque no consumieron nuevos servicios en este período, no obstante en esos proyectos prevén que con la aplicación de la guía se reduzca el tiempo de gestión de servicios.

Proyecto	Tiempo de gestión de servicios antes	Tiempo de gestión de servicios después
Activos Fijos Tangibles	7	0
Capital Humano	7	5
Estructura y Composición	10	0
Finanzas	13	4
Logística	15	7
Configuración	5	2
Costos y Proceso	15	0
Media	10,28571429	2,571428571
Varianza	16,9047619	7,952380952
Desviación estándar	4,215052392	2,828427125

Tabla 2 Tiempo promedio de gestión de servicios

En el caso de la desviación estándar es alta en el primer período, lo que expone una vez más la diferencia entre los proyectos. En el segundo período aunque sigue siendo alta se reduce considerablemente, lo que muestra que se van asemejando los tiempos en los proyectos.

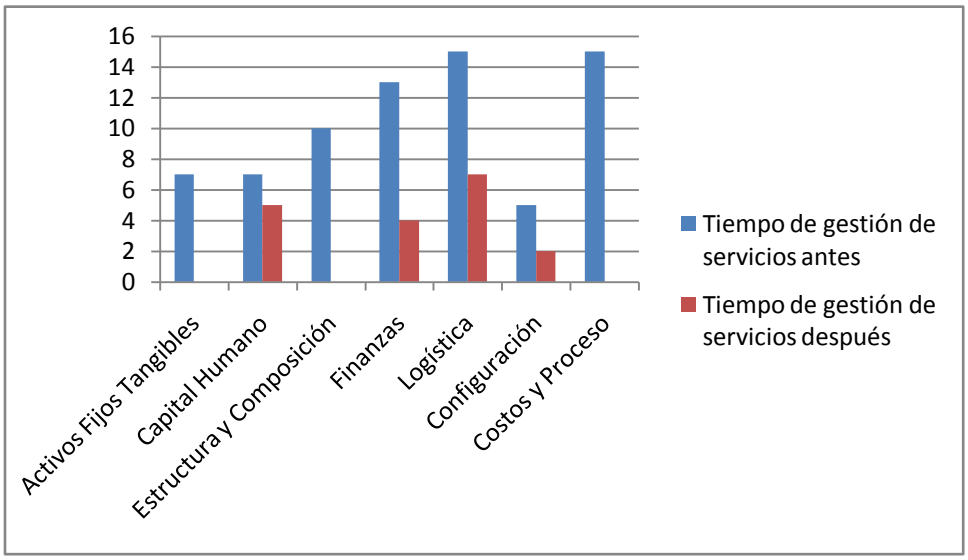


Figura 14 Tiempo promedio de gestión de servicios

Analizando los datos obtenidos en el indicador índice de desviaciones, el cual muestra el índice de ocurrencia de desviaciones por servicios consumidos en un proyecto demuestran que los resultados de aplicación de la guía fueron positivos. No obstante en este indicador no se alcanzan resultados tan significativos como en otros, lo que puede estar influenciado por el hecho de que no existe un correcto mecanismo de registro de desviaciones en los proyectos. En la Tabla 3 se muestran los resultados.

Proyecto	Índice de desviaciones antes	Índice de desviaciones después
Activos Fijos Tangibles	0,025	0
Capital Humano	0	0
Estructura y Composición	1	0
Finanzas	0,2	0,1
Logística	0,05	0,02
Configuración	0	0
Costos y Proceso	0,166666667	0
Media	0,205952381	0,017142857

Varianza	0,128998016	0,001390476
Desviación estándar	0,383611011	0,04

Tabla 3 Índice de desviaciones

Un análisis de la desviación estándar muestra que en el segundo período se reduce considerablemente, lo que evidencia la tendencia a asemejar el proceso en los proyectos, que en este caso es a reducir el índice de desviaciones por servicio consumido. En la Figura 15 se muestra una gráfica que ayuda a observar mejor el comportamiento del indicador entre los dos períodos estudiados.

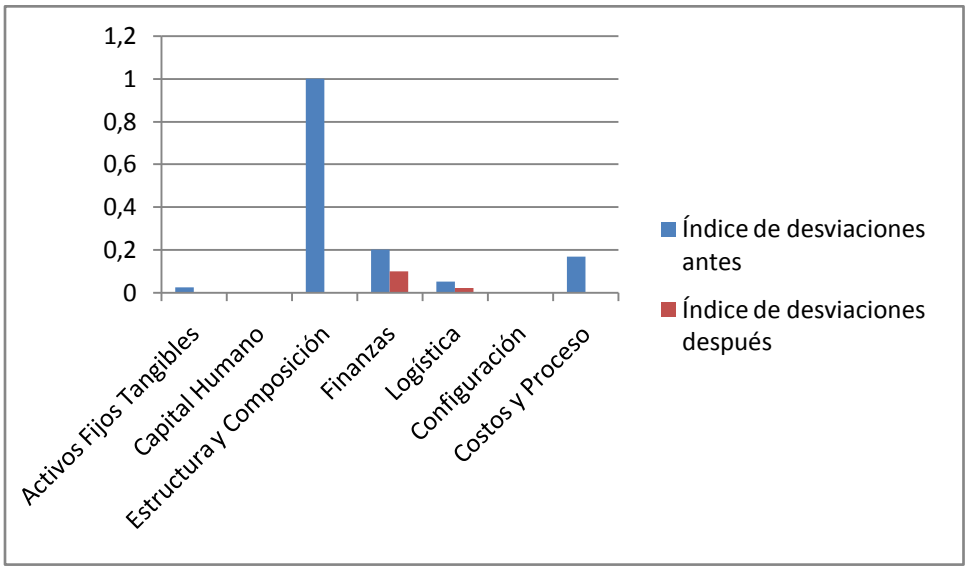


Figura 15 Índice de desviaciones

En el caso del indicador tiempo promedio de gestión de no conformidad brinda una vista del estado de las comunicaciones en los proyectos, de la importancia otorgada a la integración, entre otros elementos. Los resultados alcanzados en este rubro fueron muy satisfactorios pues en cinco de los siete proyectos se redujo este tiempo, evidenciando un impacto positivo de la aplicación de la guía, en alguno de los proyectos el tiempo se redujo más de un 50 %. La desviación estándar también se redujo casi en un 50 % demostrando la tendencia a igualar los datos en los proyectos, que en este caso tienden a reducirse. En la Tabla 4 se muestran los resultados.

Proyecto	tiempo de gestión de NC antes	tiempo de gestión de NC después
Activos Fijos Tangibles	7	0
Capital Humano	7	5
Estructura y Composición	7	0
Finanzas	7	3
Logística	15	7
Configuración	2	1
Costos y Proceso	15	7
Media	8,571428571	3,285714286
Varianza	22,61904762	9,571428571
Desviación estándar	5,154286242	2,994439291

Tabla 4 Tiempo promedio de gestión de no conformidades de integración

En la Figura 16 se muestra claramente de forma gráfica la diferencia del indicador en los dos períodos.

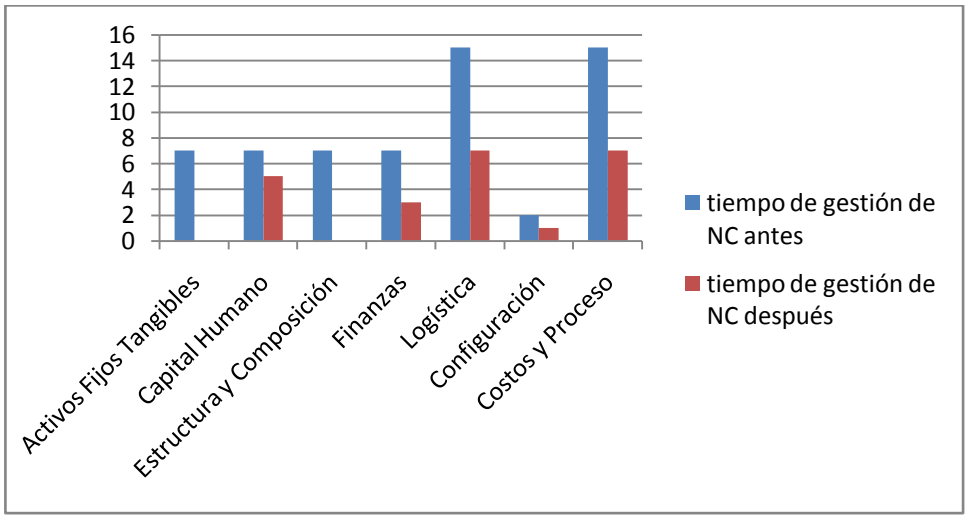


Figura 16 Tiempo promedio de gestión de no conformidades de integración

El índice de no conformidad por servicios brinda una medida de la calidad de los servicios que consume cada proyecto, mientras menor sea este índice indicará mejor calidad. Los resultados mostrados en este rubro en la Tabla 5 fueron también satisfactorios, en todos los casos se redujo el índice, aunque en algunos proyectos como no tuvieron nuevos servicios en el segundo período no se puede demostrar el impacto positivo, no obstante en todos los casos señalan que existe gran probabilidad de que se reduzca este índice.

Proyecto	Índice de no conformidades por servicios consumidos antes	Índice de no conformidades por servicios consumidos después
Activos Fijos Tangibles	0,4	0
Capital Humano	1	0,142857143
Estructura y Composición	1,666666667	0
Finanzas	0	0
Logística	0,5	0,3
Configuración	0,3	1
Costos y Proceso	0,555555556	0
Media	0,631746032	0,206122449
Varianza	0,298783069	0,135500486
Desviación estándar	0,588224643	0,390751935

Tabla 5 Índice de no conformidades por servicios consumidos

La Figura 17 ilustra gráficamente el contraste en este indicador en ambos períodos.

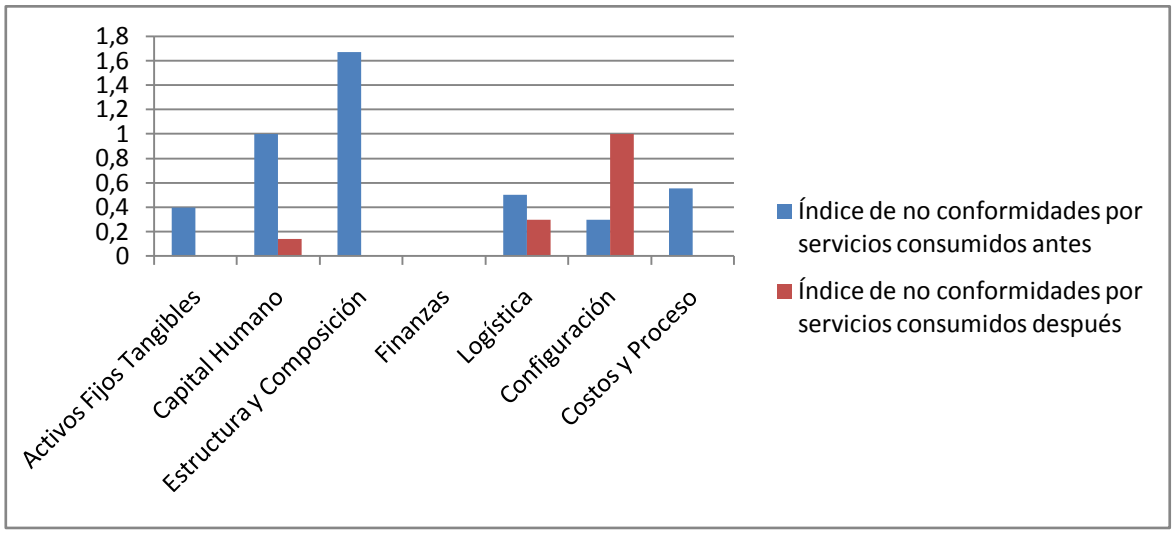


Figura 17 Índice de no conformidades por servicios consumidos

Al igual que el indicador analizado anteriormente, el índice de no conformidad por servicios brindados ofrece una medida de la calidad de los servicios brindados por un proyecto. En este caso lo resultados fueron satisfactorios en general, no obstante en uno de los proyectos el índice aumentó, lo que puede estar determinado porque se

incluyeron no conformidades a servicios implementados en el primer período. Los resultados se exponen en la Tabla 6.

Proyecto	Índice de no conformidades por servicios brindados antes	Índice de no conformidades por servicios brindados después
Activos Fijos Tangibles	0	0
Capital Humano	1	0,333333333
Estructura y Composición	0,16666667	0
Finanzas	1,11111111	0
Logística	0,5	0
Configuración	0,07246377	0,4
Costos y Proceso	0,41860465	0
Media	0,46697803	0,104761905
Varianza	0,1943912	0,032380952
Desviación estándar	0,42706767	0,190515869

Tabla 6 Índice de no conformidades por servicios brindados

En la Figura 18 se expone gráficamente la comparación del comportamiento del indicador en ambos períodos.

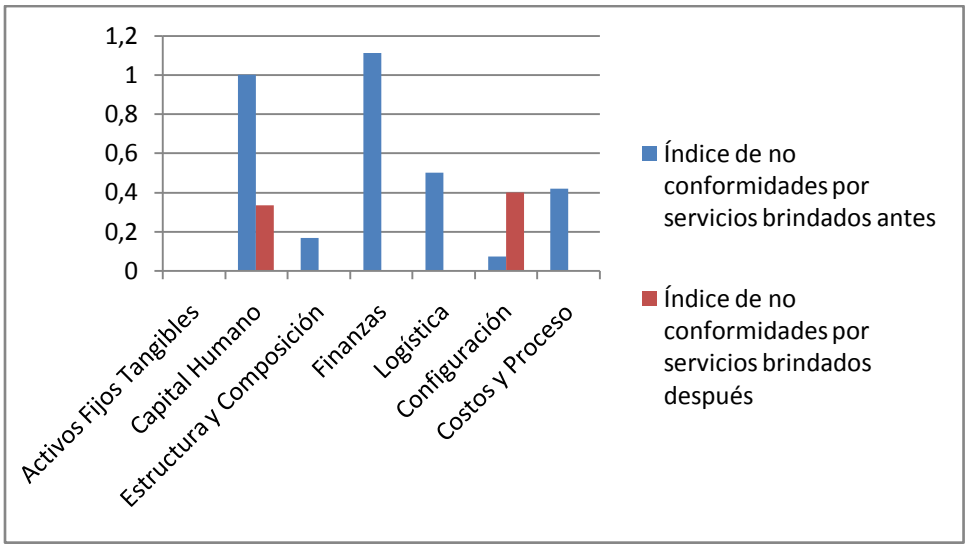


Figura 18 Índice de no conformidades por servicios brindados

Para un análisis más riguroso de comparación de las muestras se utilizó el Mann-Whitney Test. No se tiene conocimiento acerca de la normalidad de los valores de la muestra, por esta razón se utiliza el Mann-Whitney Test

para la comparación de estas dos muestras independientes. En ambos casos para el nivel de significación se aplicó el método de Monte Carlo con intervalos de confianza del 99% y se considera significativa una significación menor de 0.05. En el procesamiento de los resultados de los experimentos se utilizó el SPSS versión 13.0. En el Anexo 5 se pueden ver algunas de las tablas resultantes de estas pruebas estadísticas.

Como resultado de la aplicación del test se comprobó que existe una diferencia significativa en los valores de casi todos los indicadores, siendo significativamente mejores los valores una vez aplicada la guía propuesta. El único indicador que no tuvo diferencia significativa fue el índice de desviaciones, lo que puede estar provocado como se explicó anteriormente porque no se tienen los mejores mecanismos en el proyecto para el registro de las desviaciones, no obstante los resultados de este indicador luego de aplicar la propuesta fueron superiores. Deberá trabajarse en el estudio e influencia de la guía en este indicador de ser posible.

3.2.3. Comprobación de la adherencia al proceso

Los datos mostrados anteriormente, aunque de forma general son satisfactorios están influenciados por el grado de implementación de lo definido en la propuesta en cada uno de los proyectos. El índice de adherencia al proceso se muestra en la Tabla 7, en el libro de procesos elaborado se explica cómo obtener el índice de adherencia al proceso en un proyecto. Estos datos demuestran que se debe continuar chequeando el cumplimiento de los elementos definidos en la guía propuesta. Entre los elementos que influyeron en el porciento de adherencia al proceso con mayor impacto se encuentra lo relacionado con las responsabilidades de los arquitectos de componentes, pues además de las responsabilidades de integración en los proyectos se les asignan otras, lo que le impide llevar a cabo de forma correcta lo definido en la propuesta. En este elemento tendrá un papel muy importante los mecanismos de detección de riesgos de integración y de envío de alertas que también forman parte de la propuesta.

Proyecto	Adeherencia al proceso
Activos Fijos Tangibles	50%
Capital Humano	60%
Estructura y Composición	50%
Finanzas	50%
Logística	50%
Configuración	40%
Costos y Proceso	50%

Tabla 7 índice de adherencia al proceso

Por la importancia que brinda en la gestión del proyecto, a continuación se realiza un análisis del comportamiento de la actualización del Redmine en relación con la integración.

No existían mecanismos automatizados para capturar los datos relacionados con el proceso de integración de componentes en los proyectos antes de comenzar a aplicar la propuesta y hubo que realizarle algunas configuraciones al Redmine. Por esta causa no se puede hablar de un registro automatizado antes de aplicar la propuesta, sino que poder recogerlos por esta vía fue el comienzo y parte de la aplicación de la propuesta. Los resultados se muestran en la Tabla 8:

Indicador	Cantidad
Tareas de integración	22
No conformidades de integración	6
Desviaciones de integración	1

Tabla 8 Datos de integración

El hecho de poder contar con la información relacionada con la integración es un aporte importante de la aplicación de la propuesta. Anteriormente, como no existían mecanismos para recoger estos datos no se tenía en cuenta el impacto que tenía la integración en el desarrollo, las estimaciones para el desarrollo de un componente eran irreales y generalmente las tareas de integración generaban desviaciones porque no se planificaban en el cronograma. La aplicación de la propuesta permite planificar estas tareas, incorporarlas al cronograma y poder gestionar su cumplimiento. Un elemento negativo fue que en los datos recogidos no se llenó en el Redmine el campo referente al esfuerzo dedicado a las tareas, lo que impide hacer un análisis atendiendo al esfuerzo, medidos en horas hombres dedicados al proceso de integración, que a su vez representa un esfuerzo que es necesario planificar de lo contrario surgirán desviaciones en el desarrollo.

La información sobre la actualización del Redmine se obtuvo directamente, a través de preguntas a los arquitectos de componentes de los proyectos. Los resultados se presentan en la Tabla 9 y demuestran que los índices de actualización de los datos relacionados con la integración son muy bajos aún.

Variable	Dimensión	Indicador	Máximo valor	Valor alcanzado
Actualización del Redmine	Tareas de integración	Actualización	20	6
	No conformidades de integración	Actualización	20	6
	Actualización de desviaciones en el Redmine	Actualización	20	4

Tabla 9 Actualización del Redmine

Además de la información relacionada con la integración se tomaron datos de: las tareas de desarrollo, del total de desviaciones en los proyectos y de no conformidades reportadas en el período. En la Tabla 10 se muestran los resultados.

Indicador	Cantidad
Tareas de desarrollo	875
No conformidades (liberación, revisión y programa de mejora revisiones)	14
Total de Desviaciones	14

Tabla 10 Datos del desarrollo

En la Figura 19 se muestra el comportamiento del número de tareas de integración en comparación con las tareas de desarrollo. Igualmente se muestra la proporción de desviaciones en general y las de integración, lo mismo para las no conformidades generadas en la integración y el resto de las no conformidades reportadas.

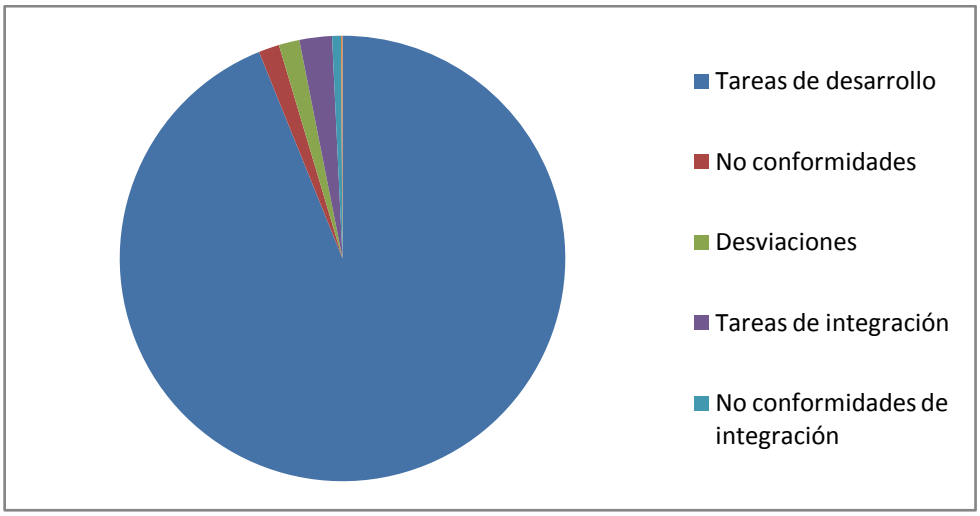


Figura 19 Proporción entre datos de integración y de desarrollo

El análisis de esta información permitirá mejorar las estimaciones y definir métricas que apoyen la toma de decisiones. Por ejemplo, del análisis realizado se muestra que por cada 16 desviaciones existe una que es por causa de la integración. Tomando en consideración estos elementos si se fuera a estimar el tiempo de desarrollo de un componente y existe algún mecanismo para estimar el tiempo de implementación, entonces el tiempo total se podría buscar haciendo una relación con el tiempo implementado estimado y el tiempo estimado para la integración a partir del análisis realizado anteriormente.

Como se pudo comprobar, uno de los factores que más influye en la adherencia al proceso definido es la actualización del Redmine. Comparando los datos obtenidos del Redmine y los capturados en el instrumento de captación de datos aplicado demuestran que todavía no existe correspondencia entre ambos datos, indicando que el índice de actualización del Redmine es bajo. Por ello y como parte del proceso de implementación se configuró en el Redmine las tareas tipo vinculadas con la integración en cada proyecto. Para ello se crearon tareas en un proyecto de prueba y luego se replicaron para el resto de los proyectos y se asignaron a los roles correspondientes. Esto permite que cada vez que se cree un nuevo proyecto ya se tengan las tareas tipos vinculadas con la integración y pueden ser incorporadas al Redmine. Además, como medida de perfección del proceso se debe continuar con la retroalimentación de los datos para lograr que mejore la estimación de las tareas vinculadas con la integración.

Como elemento significativo está el hecho de que se logró automatizar parte del proceso de integración a través de los módulos desarrollados para la gestión de servicios, el uso de estos agiliza el trabajo de integración, lo hace más confiable porque reduce la posibilidad de cometer errores humanos, además la integración de estos módulos con el Redmine permite que la información llegue por la vía correspondiente de forma rápida y directa facilitando así la gestión de proyecto.

3.3 Evaluación de la propuesta

La evaluación de la propuesta se realizó mediante una encuesta a personas con conocimientos en la gestión de proyectos que pueden hacer una valoración de la calidad de la guía, para su selección se tuvo en cuenta que llevaran más de dos años dirigiendo proyectos del centro, en total se seleccionaron 8 personas. La encuesta aplicada para lo obtención de los datos se encuentra en el Anexo 2. Los elementos evaluados en la encuesta fueron:

- Calidad de las actividades definidas.
- Evaluación de los artefactos definidos.
- Capacidad de las herramientas de apoyo.
- Evaluación de los mecanismos de apoyo a la toma decisiones.
- La evaluación de la calidad de las actividades definidas se realizó mediante la valoración de los siguientes criterios:
- Claridad.
- Completitud.
- Reusabilidad.

En cuanto a la evaluación de los artefactos se realizó mediante la valoración de los siguientes criterios:

- Claridad.
- Completitud.
- Adaptación al modelo de desarrollo.

Por su parte la capacidad de las herramientas de apoyo se realizó mediante la valoración de los siguientes criterios:

- Nivel de integración entre las herramientas.
- Utilidad.

Para la evaluación de los mecanismos para el apoyo a la toma de decisiones se realizó mediante la valoración de los siguientes criterios:

- Validez.
- Utilidad.
- Capacidad de detección automatizada de riesgos.

Criterio	Elementos de evaluación	Puntuación	Total	% Con respecto al total
Calidad de las actividades	❖ Claridad	16	16	100 %
	❖ Completitud	16	16	100 %
	❖ Reusabilidad	12	16	75 %
Evaluación de los artefactos	❖ Claridad	16	16	100 %
	❖ Completitud	15	16	94%
	Adaptación al modelo de desarrollo	13	16	81%
Capacidad de las herramientas de apoyo	❖ Nivel de integración entre las herramientas	12	16	75 %
	❖ Utilidad	12	16	75 %
Evaluación de los mecanismos para el	❖ Validez	16	16	100%
	❖ Utilidad	14	16	88%

apoyo a la toma de decisiones	❖ Capacidad de detección automatizada de riesgos	13	16	81 %

Tabla 11 Resultados de la aplicación de la encuesta

De los once criterios, todos obtuvieron más del 75 % de la puntuación. Los criterios con resultados más bajos fueron: reusabilidad de las actividades, nivel de integración entre las herramientas y la utilidad de las herramientas de apoyo. En el caso de la reusabilidad de las actividades puede estar dada esta puntuación porque las actividades que se proponen tienen un alto nivel de especificidad para la integración en los proyectos del sistema CEDRUX lo que determina que sea menos probable su reutilización en otros contextos. La puntuación del nivel de integración entre las herramientas y su utilidad puede estar dada porque todavía es necesario incorporarle nuevas funcionalidades a las herramientas y lograr que tengan un mayor nivel de integración que permita que la información se pueda obtener en cualquier momento desde alguna de ellas.

En la Figura 20 se muestra el valor obtenido por los indicadores que evalúan la calidad de las actividades propuestas y el total que podían alcanzar.

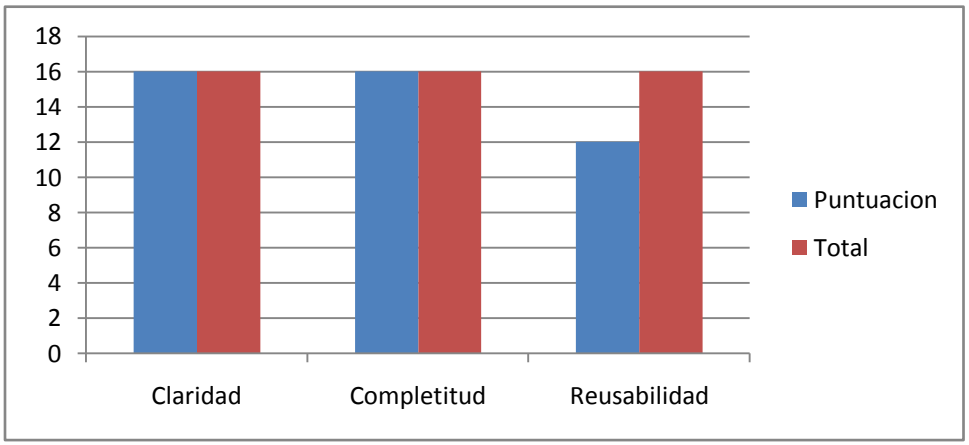


Figura 20 Indicadores de calidad de las actividades

Un análisis descriptivo de estos indicadores mostrados en la Tabla 12 permite conocer que la media fue mayor que 1.5, demostrando que los indicadores alcanzaron una alta puntuación.

El valor más bajo lo obtuvo el indicador Reusabilidad, lo que puede estar causado porque las actividades propuestas son específicas de la integración en el proyecto y puede que no tengan la misma utilidad en otro contexto.

Criterio	Min	Max	Puntuación	Media	Varianza	Desviación estándar
Claridad	2	2	16	2	0	0
Completitud	2	2	16	2	0	0
Reusabilidad	1	2	12	1,5	0,285714286	0,53452248

Tabla 12 Análisis descriptivo de los indicadores de la calidad de las actividades

Otro elemento a analizar es la desviación estándar, que en este caso menos en uno de los indicadores fue de cero, lo que demuestra un bajo índice de dispersión en la opinión de los encuestados. En la Figura 21 se muestra el valor obtenido por los indicadores para la evaluación de los artefactos propuestos y el total que podían alcanzar.

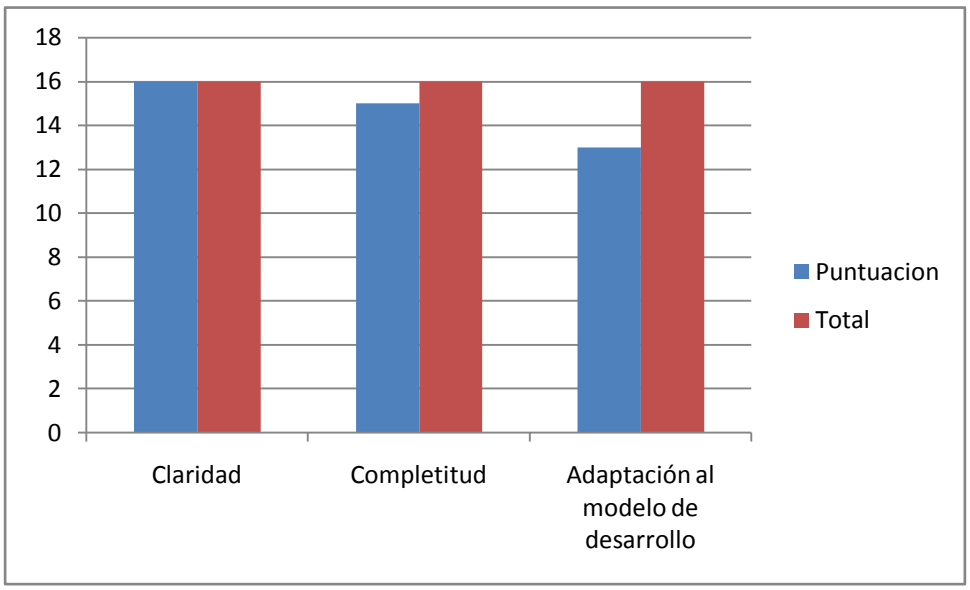


Figura 21 indicadores de evaluación de los artefactos

Un análisis descriptivo de los indicadores mostrados en la Tabla 13 permite conocer que la media fue mayor que 1.625, lo que demuestra la alta puntuación alcanzada en todos los indicadores.

Criterio	Min	Max	Puntuación	Media	Varianza	Desviación estándar
Claridad	1	2	16	2	0	0
Completitud	1	2	15	1,875	0,125	0,35355339
Adaptación al modelo de desarrollo	1	2	13	1,625	0,214285714	0,267857143

Tabla 13 Análisis descriptivo de los indicadores de evaluación de los artefactos

Para esta dimensión la desviación estándar en general fue baja, demostrando un bajo índice de dispersión en la opinión de los encuestados.

En la Figura 22 se muestra el valor obtenido por los indicadores para evaluar las herramientas de apoyo al proceso.

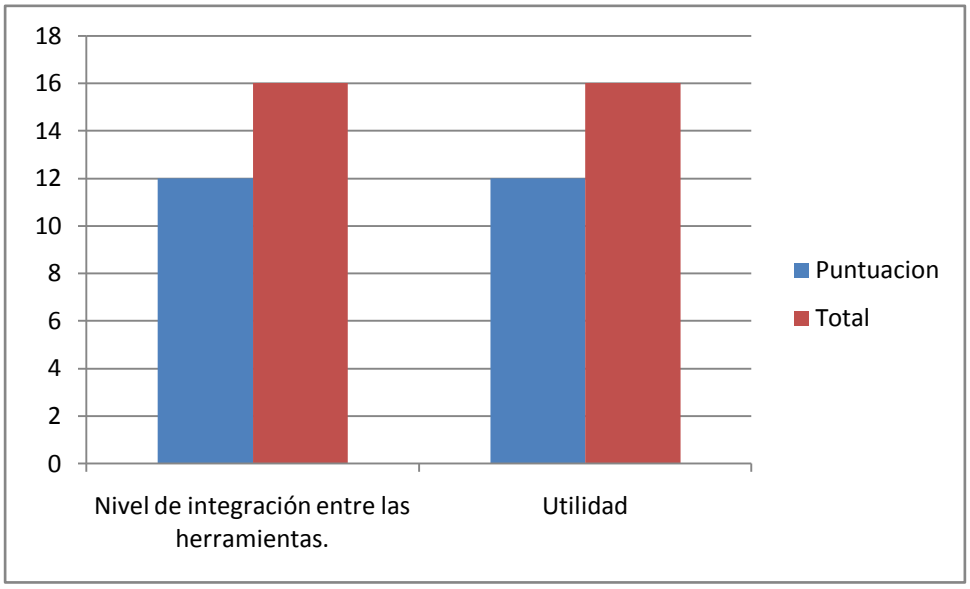


Figura 22 Indicadores de evaluación de las herramientas de apoyo

Un análisis descriptivo de los indicadores mostrados en la Tabla 14 permite conocer que la media fue de 1.5 en ambos indicadores, demostrando una aceptable puntuación. Comparada con otros indicadores fue más baja, la causa puede estar en que las herramientas no se han probado lo suficiente para que demuestren sus capacidades.

Criterio	Min	Max	Puntuación	Media	Varianza	Desviación estándar
Nivel de integración entre las herramientas	0	2	12	1,5	0,571428571	0,75592895
Utilidad	1	2	12	1,5	0,571428571	0,75592895

Tabla 14 Análisis descriptivo de los Indicadores de evaluación de las herramientas de apoyo

En este caso la desviación estándar si fue alta, en ambos casos de 0.75 lo que demuestra un alto índice de dispersión en la opinión de los encuestados.

En la Figura 23 se muestra el valor obtenido por los indicadores para evaluar las herramientas de apoyo al proceso.

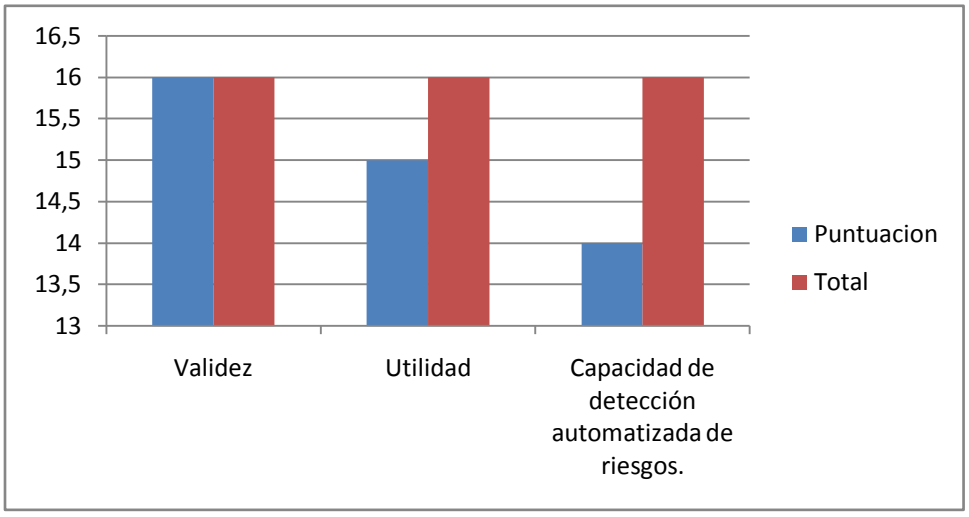


Figura 23 Indicadores de evaluación de los mecanismos para el apoyo a la toma de decisiones

Un análisis descriptivo de los indicadores mostrados en la Tabla 15 permite conocer que la media fue mayor que 1,75, lo que demuestra la alta puntuación alcanzada por los indicadores en esta dimensión.

Criterio	Min	Max	Puntuación	Media	Varianza	Desviación estándar
Validez	2	2	16	2	0	0

Utilidad	1	2	14	1,75	0,214285714	0,46291005
Capacidad de detección automatizada de riesgos.	1	2	14	1,625	0,267857143	0,51754917

Tabla 15 Análisis descriptivo de los Indicadores de evaluación de los mecanismos para el apoyo a la toma de decisiones

La desviación estándar sólo superó los 0.5 en el tercer indicador, lo que demuestra un bajo índice de dispersión en los valores obtenidos.

Realizando un balance general de la evaluación, en la Figura 24 se muestra la puntuación total obtenida en cada una de las dimensiones en las que se evaluó la guía.

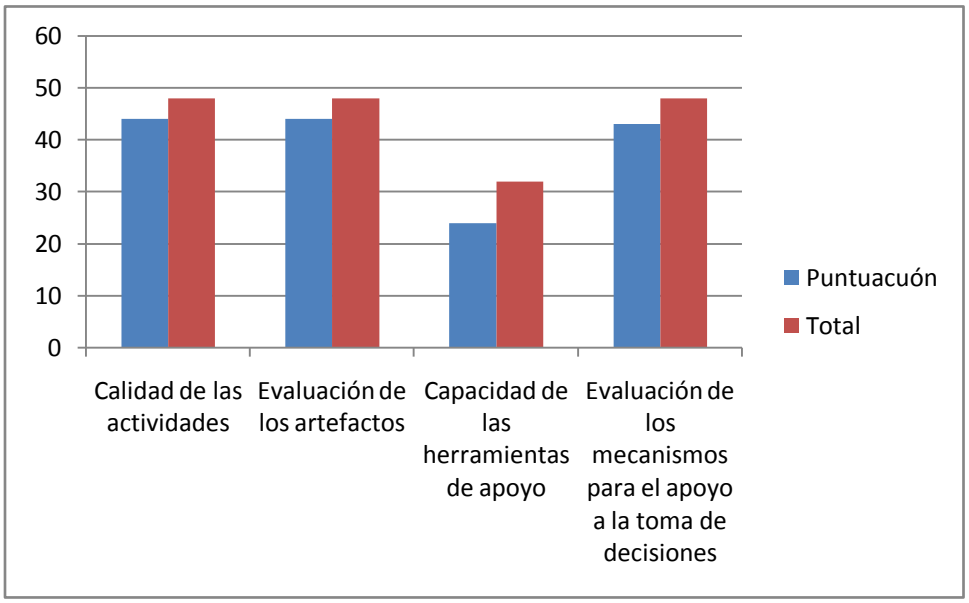


Figura 24 Puntuación de los criterios generales

En la Tabla 16 se muestran todos los indicadores ordenados descendentemente por el valor alcanzado en la media. Como se puede apreciar los indicadores con mejores resultados fueron: claridad y completitud de las actividades, claridad de los artefactos y la validez de los mecanismos de detección de riesgos. La media de estos indicadores fue 2, que es la máxima puntuación posible. Demostrando que en estos indicadores se obtuvo buena valoración por parte de los encuestados.

En general los resultados fueron buenos, todos los indicadores obtuvieron puntuaciones con una media superior a 1,5. Los niveles de consenso fueron altos en casi todos los casos, lo que demuestra que se seleccionó el personal adecuado a encuestar.

Criterio	Puntuación	Total	Media	Varianza	Desv. stan.
Claridad de las actividades	16	16	2	0	0
Completitud de las actividades	16	16	2	0	0
Claridad de los artefactos	16	16	2	0	0
Validez de los mecanismos de detección de riesgos	16	16	2	0	0
Completitud de los artefactos	15	16	1,875	0,125	0,35355339
Utilidad de los mecanismos de detección de riesgos	14	16	1,75	0,21428571	0,46291005
Adaptación al modelo de desarrollo de los artefactos	13	16	1,625	0,26785714	0,51754917
Capacidad de detección automatizada de riesgos.	13	16	1,625	0,26785714	0,51754917
Reusabilidad de las actividades	12	16	1,5	0,28571429	0,53452248
Nivel de integración entre las herramientas.	12	16	1,5	0,57142857	0,75592895
Utilidad de las herramientas	12	16	1,5	0,57142857	0,75592895

Tabla 16 Análisis descriptivo de todos los indicadores

3.4 Conclusiones del capítulo

- La ejecución del experimento realizado permitió conocer el impacto positivo que tiene la aplicación de la propuesta en la eficiencia del proceso de desarrollo en los proyectos. Todos los indicadores mejoraron con la aplicación de la propuesta, destacándose la reducción del tiempo promedio de implementación de servicios, el tiempo de gestión de servicios y el tiempo de gestión de las no conformidades de integración. Muestra de lo expresado anteriormente está el hecho de que en algunos de los proyectos estos tiempos llegaron a reducirse hasta en un cincuenta por ciento.
- La evaluación de la propuesta a partir de los indicadores definidos arrojó resultados positivos en todos los indicadores, demostrando que la guía cumple con los parámetros propuestos en cada una de las dimensiones. Los indicadores más destacados fueron: claridad y completitud de las actividades, claridad de los artefactos y la validez de los mecanismos de detección de riesgos. Mientras que los de resultados más discretos fueron: reusabilidad de las actividades, nivel de integración entre las herramientas propuestas y la utilidad de las herramientas propuestas. En estos indicadores aunque no se alcanzaron valores tan altos como en el resto, la puntuación fue buena pues alcanzaron el setenta y cinco porciento del total.

Conclusiones Generales

- La definición del marco teórico de la investigación permitió conocer los principales elementos teóricos vinculados con la integración de componentes. Se pudo investigar los principales referentes teóricos del tema en cuestión y obtener los elementos teóricos bases de la propuesta realizada. Se pudo determinar las dos principales tendencias con las que se ejecuta la integración de componentes, la primera es desarrollar la integración sólo cuando hallan sido implementados los componentes y la segunda es comenzar a integrar desde el inicio y durante todo el proceso de implementación. La primera tendencia es la que se propone en las metodologías tradicionales como RUP y la segunda es la propuesta por las metodologías ágiles. Se pudo demostrar que por la características del proyecto es necesario efectuar la integración continua, no obstante se deben tener en cuenta elementos definidos en las metodologías tradicionales, entre los que se encuentran los roles, artefactos y actividades específicas para desarrollar la integración. El estudio de experiencias en proyectos nacionales permitió conocer iniciativas que se llevan a cabo, entre ellas resalta lo aplicado en el centro GEITEL, donde la utilización de la metodología SXP y el Test de Joel le ha permitido mejorar considerablemente el proceso de integración. No obstante en el plano nacional de manera general todavía no se le da la importancia requerida al tema y sólo se limitan a seguir lo definido en la metodología de desarrollo que utilizan.
- La guía propuesta, realizada a partir del estudio de los elementos teóricos estudiados y de las experiencias prácticas contribuirá a mejorar la eficiencia del proceso de desarrollo en los proyectos del sistema CEDRUX. En la propuesta se incluyen: el flujo de actividades a seguir en la integración, los roles de mayor responsabilidad, los artefactos, los lineamientos que se deben cumplir y las herramientas automatizadas que se deben utilizar, entre otros elementos.
- La aplicación de la guía permitió comprobar su comportamiento desde el punto de vista práctico, arrojando resultados positivos y demostrando que su aplicación puede contribuir a mejorar el proceso de desarrollo en los proyectos del sistema CEDRUX. Muestra de ello fueron los resultados alcanzados en los indicadores definidos para medir la eficiencia del proceso de desarrollo los cuales mejoraron con la aplicación de la guía, destacándose los indicadores tiempo promedio de implementación de un servicio y el tiempo promedio de gestión de un servicio. Los buenos resultados en la aplicación de la propuesta también se ratificaron con el test de Mann-Whitney. La evaluación de la propuesta a partir de los indicadores definidos arrojó resultados positivos

en todos los indicadores, demostrando que la guía cumple con los parámetros propuestos en cada una de las dimensiones. Los indicadores más destacados fueron: claridad y completitud de las actividades, claridad de los artefactos y la validez de los mecanismos de detección de riesgos. Mientras que los de resultados más discretos fueron: reusabilidad de las actividades, nivel de integración entre las herramientas propuestas y la utilidad de las herramientas propuestas. En estos indicadores aunque no se alcanzaron valores tan altos como en el resto, la puntuación fue buena pues alcanzaron el setenta y cinco porciento del total.

Recomendaciones

- Continuar aplicando la guía propuesta e ir perfeccionándola teniendo en cuenta los resultados prácticos que vaya obteniendo y del momento y las condiciones en que se encuentren los proyectos.
- Desarrollar los pluggin propuestos al Redmine.
- Continuar desarrollando las herramientas propuestas para el apoyo al proceso de integración y mejorar la integración entre todas ellas.
- Incorporar un estándar para lograr la generisidad de los servicios y evitar su repetición.
- Realizar cambios a la propuesta para generalizarla y que pueda ser reutilizada en otros proyectos.

Bibliografía

Beck, K., M. Beedle, et al. (2001). "Manifiesto por el Desarrollo Ágil de Software." 2010, from http://agilemanifesto.org/.

Beck, K. C. A. (2009). " Extreme Programming Explained, Embrace Change - ", from http://www.addison-wesley.de/0321278658.html

Brooks, F. P. J. (1995). The Mythical Man-Month. University of North Carolina at Chapel Hill, ADDISON-WESLEY.

Bustos, G. A., Ricardo . (2003). "METODOS DE DESARROLLO DE SOFTWARE: EL DESAFIO PENDIENTE DE LA ESTANDARIZACION. (Spanish)." SOFTWARE DEVELOPMENT METHODOLOGIES: A DUEL PENDING FOR STANDARDIZATION. (English) **12**: 23-42.

Canós, J. H. L., Patricio , Penadés, M Carmen (2005). "Métodologías Ágiles en el Desarrollo de Software."

Carlos Menezes. (2003). "www.cmenez.wordpress.com." Retrieved 2009, from http://cmenez.wordpress.com/2007/02/19/erps-planificacion-de-recursos-empresariales/.

Clements, P. N., Linda (2002). Sofware Product Lines : Practice and Patterns, Addison Wesley.

Clements, P. S., Mary (1996). "A field guide to Boxology: Preliminary classification of architectural styles for software systems."

Cockburn, A. (2001). Agile Software Development.

Duvall, P. (2009). " Continuous Integration anti-patterns." Automation for the people.

Duvall, P. S. M., Andrew. (2007). "Continuous Integration, Improving Software Quality and Reducing Risk -." from http://www.addison-wesley.de/main/main.asp?page=englisch/bookdetails&ProductID=121548

Dybá, T. D., Torgeir (2008). "Empirical studies of agile software development: A systematic review." Information and Software Technology **50**(9-10): 833-859.

Entregables, D. (2010). "Documentos entregables".

Fleischer, G. (2009). "Continuous Integration. What companies expect and solutions provide." Fontys.

Fowler, M. (2006). "Continuous Integration."Retrieved 06/05/2010, 2010, from http://martinfowler.com/articles/continuousIntegration.html

González, R. A. H. L. S. C. (2002). EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA. Ciudad de la Habana EDUNIV.

Kiccillof, N. R., Carlos (2005). Estilos y Patrones en la Estrategia deArquitectura de Microsoft Versión 1.0. UNIVERSIDAD DE BUENOS AIRES.

Krueger, C. W. (2010). "Introduction to Software Product Lines." Retrieved 18/05, 2010, from http://softwareproductlines.com/.

Larsson, S. (2005). IMPROVING SOFTWARE PRODUCT INTEGRATION. Department of Computer Science and Engineering Sweden, Mälardalen University.

Martínez, L. F. I. (2003). "Un Modelo de Mediación para el desarrollo de software basado en Componentes COTS."

Microsoft, C. (2005). "MSF for Agile Software Development 4.2."

Microsoft., C. (2003). "Microsoft Solutions Framework 3.0."

Montilva, J. A. (2006). "Desarrollo de Software Basado en Líneas de Productos de Software." IEEE Computer Society Región 9.

Montilva, J. A. C., N. Arapé, et al. (2005). "Desarrollo de Software Basado en Componentes."

Pressman, R. S. (2001). Ingeniería del sofware . Un enfoque práctico. Habana , Cuba.

Rational, C. S. (2003). Rational Unified Process.

Rodríguez, G (2008).”Metodología ágil para proyectos de software libre."

SEI, S. E. I. (2008). "Software Engennering Institute." 2008, from http://www.sei.cmu.edu/.

Sommerville, I. (2005). Ingeniería del Software, Addison-Wesley.

Szyperski, C. (1998). "Component Software. Beyond Object-Oriented Programming."

Wallnau, K. B., Alan (1998). “The current state of CBSE”. IEEE Software.

Anexos

Anexo 1. Entrevista a roles importantes de los proyectos

Nombre de la persona: _____

Tiempo de experiencia en la gestión de proyectos: _____

Rol: _____

(Jefes de línea, Jefe de departamento de desarrollo de productos, Arquitecto de sistema, Arquitecto de proyecto, Desarrollador)

1. ¿En cuantos proyectos ha trabajado o dirigido de una forma directa o indirecta?
2. ¿En los proyectos en que ha trabajado se han presentado atrasos en la entrega final de los productos o en las fases intermedias?
3. ¿La integración de los componentes es una de las razones?
4. Seleccione de los siguientes aspectos los que usted considere que hayan provocado atraso en el proyecto:

   ___ El tiempo dedicado a la integración fue mayor del planificado

   _____Los servicios consumidos no poseen la calidad requerida

   _____Demora en la implementación de nuevos servicios requeridos por el proyecto

   _____La demanda de servicios es muy alta

   _____La tecnología utilizada para la integración es muy compleja

   _____Otras causas:

5. ¿Usted considera que el proceso de integración se desarrolla de forma organizada?
6. ¿Las personas que llevan a cabo la integración de componentes en los proyectos son las adecuadas?
7. ¿Según su valoración cuales cree que deberían ser las competencias de los arquitectos y desarrolladores que trabajen en la integración?

8. ¿Se sigue alguna guía formal, procedimiento o metodología para gestionar la integración de componentes?
9. ¿Considera que sería útil definir algunos de estos elementos para organizar la integración de componentes en los proyectos?
10. ¿Es adecuada la planificación del orden de desarrollo e integración de componentes?
11. ¿Son incorporadas al cronograma de los proyectos las tareas relacionadas con la integración de componentes?
12. ¿Se le da seguimiento a las tareas relacionadas con la integración de componentes?
13. ¿Existen mecanismos para evaluar la calidad de los servicios brindados por un componente?
14. ¿Existen mecanismos para recoger las incidencias relacionadas con la integración de componentes?
15. ¿Cuando existe problema con algún servicio, se resuelve rápidamente?
16. ¿Por causa de la integración se añaden desviaciones?
17. ¿Qué artefactos usted considera se deben tener en cuenta para organizar el proceso de integración de componentes?
18. Si se le incorporara al Redmine la funcionalidad que le alerte cuando existen irregularidades con la integración. ¿Cuándo usted desearía que le avisara y qué información?
19. ¿A su consideración qué validaciones podría tener el Redmine que ayuden a evitar problemas relacionados con la integración posteriormente?
20. ¿Existen otros elementos que usted considera se deberían tener en cuenta para la organización del proceso de integración?

Anexo 2. Evaluación de calidad de la guía de acciones

Nombre de la persona: _____

Años de experiencia en la gestión de proyectos: _____

(Expertos en gestión de proyectos)

Evaluación de las actividades

1. Realice una evaluación de las actividades definidas en cuanto a la claridad en las definiciones y explicaciones:

   Alta___ Media___ Baja___

2. Según su valoración, la completitud de las actividades definidas es:

   Alta___ Media___ Baja___

3. Clasifique la reusabilidad en otros contextos de las actividades propuestas en:

   Alta___ Media___ Baja___

Evaluación de los artefactos

4. Realice una evaluación de las actividades definidas en cuanto a la claridad en las definiciones y explicaciones:

   Alta___ Media___ Baja___

5. Según su valoración, la completitud de las actividades definidas es:

   Alta___ Media___ Baja___

6. Considera que la adaptabilidad de los artefactos definidos al modelo de desarrollo es:

   Alta___ Media___ Baja___

Evaluación de las herramientas de apoyo al proceso

7. Considera que el nivel de integración entra las herramientas definidas para el apoyo al proceso de integración de componentes es :

   Alto___ Medio___ Bajo ___

8. Según su valoración, la utilidad de las herramientas definidas para el apoyo al proceso de integración de componentes es:

   Alta___ Media___ Baja___

Evaluación de los mecanismos de apoyo a la toma de decisiones

9. Realice una valoración de la validez que tienen las reglas definidas para la generación automática de alertas en :

   Alta___ Media___ Baja___

10. Según su valoración, la utilidad de las alertas automáticas definidas es:

    Alta___ Media___ Baja___

11. Considera que la Capacidad de detección automatizada y generación de riesgos de integración es:

    Alta___ Media___ Baja___

12. Considera que en la guía hay elementos que no estén reflejados y sean de vital importancia para mejorar el proceso de integración de componentes.

    Si___ No___

    ¿Podría mencionarlos?

    _____

    _____

    _____

Anexo 3. Encuesta para conocer experiencias en los proyectos nacionales

Centro: _____

Rol: _____

1. ¿Cuántos proyectos desarrolla su centro (o departamento)?
2. ¿Qué metodología de desarrollo se aplica en su centro?
3. ¿Para gestionar la integración de componentes en los proyectos, se utiliza lo definido en la metodología de desarrollo o tienen definido algún procedimiento específico?
    a. En caso que si utilice un procedimiento específico, explíquelo brevemente.
4. Alguno de los proyectos han sufrido atrasos por causa de la integración de componentes.
5. ¿Existen otros elementos que usted considera relevantes sobre la organización del proceso de integración?

Anexo 4. Instrumento para la captación de datos de integración

Nombre de la persona: _____

Proyecto: _____

Rol: _____

Periodo: _____ (Inicial o Final)

1. ¿Cuántos días como promedio se demora en la implementación de un servicio?
2. ¿Cuántos días transcurren como promedio desde que se detecta la necesidad de un nuevo servicio hasta que le notifican que ya está disponible?
3. ¿Cuántas desviaciones ha tenido por causas de integración?
4. ¿Cuántas no conformidades se le han notificado sobre los servicios que brinda su proyecto?
5. ¿Cuántas no conformidades ha detectado en los servicios que consume su proyecto?
6. ¿Cuántos días transcurren como promedio desde que se detecta una no conformidad en un servicio que consume su proyecto hasta que le notifican que ya está resuelta?
7. En el caso que no halla tenido integraciones en el periodo. ¿Considera que con la aplicación de la guía metodológica para gestionar la integración de componentes los anteriores indicadores mejoren?

Anexo 5. Resultados de la aplicación del test de Mann-Whitney

NPar Tests

Descriptive Statistics					
	N	**Mean**	**Std. Deviation**	**Minimum**	**Maximum**
Tiempo de desarrollo de servicio	14	2,50	2,210	0	7
Tiempo de gestión de servicios	14	6,43	5,244	0	15
Índice de desviaciones	14	,11155	,264154	,000	1,000
Índice de no conformidades por servicios brindados	14	,29	,374	0	1
Índice de no conformidades por servicios consumidos	14	,419	,4992	,0	1,7
tiempo de gestión de NC	14	5,93	4,731	0	15
muestras	14	1,50	,519	1	2

Mann-Whitney Test

Ranks				
	muestras	**N**	**Mean Rank**	**Sum of Ranks**
Tiempo de desarrollo de servicio	**1**	7	10,21	71,50
	2	7	4,79	33,50
	Total	14		
Tiempo de gestión de servicios	**1**	7	10,64	74,50
	2	7	4,36	30,50
	Total	14		
Índice de desviaciones	**1**	7	9,43	66,00
	2	7	5,57	39,00
	Total	14		

Índice de no conformidades por servicios brindados	**1**	7	9,79	68,50
	2	7	5,21	36,50
	Total	14		
Índice de no conformidades por servicios consumidos	**1**	7	9,57	67,00
	2	7	5,43	38,00
	Total	14		
tiempo de gestión de NC	**1**	7	9,86	69,00
	2	7	5,14	36,00
	Total	14		

Test Statistics(c)

			Tiempo de desarrollo de servicio	Tiempo de gestión de servicios	Índice de desviaciones	Índice de no conformidades por servicios brindados	Índice de no conformidades por servicios consumidos	tiempo de gestión de NC
Mann-Whitney U			5,500	2,500	11,000	8,500	10,000	8,000
Wilcoxon W			33,500	30,500	39,000	36,500	38,000	36,000
Z			-2,542	-2,842	-1,842	-2,128	-1,899	-2,200
Asymp. Sig. (2-tailed)			,011	,004	,065	,033	,058	,028
Exact Sig. [2*(1-tailed Sig.)]			,011(a)	,002(a)	,097(a)	,038(a)	,073(a)	,038(a)
Monte Carlo Sig. (2-tailed)	Sig.		,014(b)	,004(b)	,084(b)	,035(b)	,067(b)	,028(b)
	99% Confidence Interval	Lower Bound	,011	,003	,077	,030	,060	,024
		Upper Bound	,017	,006	,091	,039	,073	,033
Monte Carlo Sig. (1-tailed)	Sig.		,008(b)	,003(b)	,041(b)	,016(b)	,032(b)	,014(b)
	99% Confidence Interval	Lower Bound	,006	,001	,036	,013	,027	,011
		Upper Bound	,010	,004	,047	,019	,036	,017

a Not corrected for ties.

b Based on 10000 sampled tables with starting seed 1314643744.

c Grouping Variable: muestra

Anexo 6. Test de Joel

- ¿Utilizas software de control de versiones?
- ¿Puedes generar el producto en un solo paso?
- ¿Compilas el producto diariamente?
- ¿Tienes una base de datos para los bugs?
- ¿Corriges los bugs antes de añadir más código?
- ¿Tienes una planificación actualizada?
- ¿Tienes un documento de especificaciones?
- ¿Están los programadores en un lugar tranquilo?
- ¿Utilizas las mejores herramientas que puedes comprar?
- ¿Tienes gente para probar los productos?
- ¿Haces escribir código a los nuevos candidatos en las entrevistas?
- ¿Haces pruebas de usabilidad "de vestíbulo"?

Glosario de Términos

Componente: Es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que puede ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski 1998).

Integración: El proceso desarrollado cuando las partes son combinadas para formar partes más complejas y finalmente en productos completos. Los elementos críticos en la integración incluyen descripción y gestión de interfaces, la secuencia en la cual los componentes son integrados y la comunicación entre diferentes involucrados. Incluso se reconoce que también se incluyen requerimientos y propiedades del sistema que no pueden ser comprobados desde un nivel de componente, pero que deben ser comprobados a nivel de sistema (Larsson 2005).

ERP: Sistema para la planificación de recursos empresariales.

CEDRUX: Denominación del sistema ERP que se desarrolla para ser usado en las empresas cubanas.