Universidad de las Ciencias Informáticas.

Facultad 3

Centro de Informatización de la Gestión de Entidades

Propuesta metodológica para la obtención de los componentes de software en los proyectos del sistema Cedrux.

Tesis en virtud de optar por el título de máster en Gestión de Proyectos Informáticos

---

Autor: Ing. Osmar Leyet Fernández.

Tutor: Msc. Yadenis Piñero Pérez

septiembre 2011

Ciudad de la Habana

A:

…la persona que me ha entregado gran parte de su vida, su cariño, dedicación y amor irán conmigo donde quiera que esté, te debo lo que soy y hasta donde llegue, gracias por la vida…

Que sirva la presente, como un pequeño homenaje a mi abuela María Porta…

DECLARACIÓN JURADA DE AUTORÍA

Yo Osmar Leyet Fernández, con carnet de identidad 84091124741, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada "Propuesta metodológica para la obtención de los componentes de software en el sistema Cedrux", para optar por el título de Máster en Gestión de Proyectos Informáticos. El presente trabajo fue desarrollado en el transcurso de los años 2009-2011. Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los __ días del mes de marzo del año 2011.

_____

Firma

RESUMEN

En el trabajo se realiza un estudio de las corrientes y modelos arquitectónicos existentes, de las tendencias actuales y de los elementos relacionados con el desarrollo de software guiado por componentes. Basado en las características de los proyectos de planificación de recursos empresariales y los modelos arquitectónicos definidos para estos proyectos, se definen una serie de elementos metodológicos y componentes arquitectónicos de alto significado para el proceso de construcción de la arquitectura de sistema.

Se propone una taxonomía de componentes y una guía metodológica para la obtención de los mismos, de manera que se incida de forma positiva en la disminución de los tiempos de desarrollo de los proyectos presentes en el programa ERP-Cuba.

Se realiza un análisis de los resultados obtenidos de la aplicación de la guía que se propone en los proyectos del programa ERP-Cuba desarrollado en la Universidad de las Ciencias Informáticas, mostrándose la efectividad de la aplicación de la misma mediante los indicadores definidos.

Palabras claves: modelos arquitectónicos, componentes, ERP.

Abstract

The paper makes a study of some existing architectural tendencies and models, current trends and items related to the components-oriented software development.

Based on the characteristics of enterprise resource planning projects and architectural models defined for them, a set of methodological elements and architectural components of high significance for the system architecture building process, are defined.

It is proposed a taxonomy of components and a methodological guide to obtain the same, as a way to have a positive impact in reducing the development time of the projects in the ERP- Cuba program.

It is done an analysis of the results of applying the guidance proposed to projects in the ERP- Cuba program, developed at the University of Information Sciences, showing the effectiveness of the implementation thereof by the defined indicators.

Keywords: *architectural models, components, ERP.*

ÍNDICE

INTRODUCCIÓN **7**

CAPÍTULO 1. FUNDAMENTO TEÓRICO **15**

1.1 Introducción **15**

1.2 Arquitectura de software **15**

1.2.1 Modelos fundamentales 15

1.2.2 Consideraciones sobre las corrientes estudiadas 17

1.3 Estilos de construcción de arquitecturas **18**

1.3.1 Estrategia de arquitectura de Microsoft 18

1.3.2 MODEL DRIVEN ARCHITECTURE (MDA) 19

1.3.3 Attribute-Driven Design (ADD) 21

1.3.4 Arquitecturas orientadas a componentes 21

1.3.5 Consideraciones sobre los estilos estudiados 22

1.4 Desarrollo de software basado en componentes **23**

1.4.1 Componentes 23

1.4.2 Interacción entre componentes 24

1.4.3 Reutilización de componentes 26

1.4.4 Consideraciones sobre los conceptos estudiados 27

1.5 Métodos para la obtención de componentes **27**

1.5.1 WATCH-COMPONENT 28

1.5.2 FODA 30

1.5.3 JODA 32

1.5.4 Consideraciones sobre los métodos estudiados 33

1.6 Conclusiones Parciales **33**

CAPÍTULO 2. PROPUESTA METODOLÓGICA **35**

2.1 Introducción **35**

2.2 Elementos metodológicos de la propuesta **35**

2.3 Modelo del producto **36**

2.3.1 Incidencia en las vistas arquitectónicas 36
2.3.2 Taxonomía de componentes 36
2.3.3 Interacciones entre componentes 39

2.4 Modelo de proceso 41
2.4.1 Organización estructural del equipo 41
2.4.2 Guía metodológica para la definición de los componentes del sistema 42

2.5 Modelo del equipo 46
2.5.1 Roles en el proceso de obtención de los componentes 46
2.5.2 Artefactos 49

2.6 Conclusiones Parciales 53

CAPÍTULO 3. ANÁLSIS DE RESULTADO 54

3.1 Introducción 54

3.2 Indicadores de medición 54
3.2.1 Características de la muestra 54

3.3 Evaluación 55
3.3.1 Análisis de reutilización 55
3.3.2 Análisis de disminución del tiempo de desarrollo 59
3.3.3 Ejecución del experimento 61

3.4 Impacto 65
3.4.1 Analisis costo beneficio 67

3.5 Conclusiones Parciales 67

CONCLUSIONES 69

RECOMENDACIONES 70

REFERENCIAS 71

ANEXOS 75

INTRODUCCIÓN

La creciente necesidad de informatización de las entidades del mundo empresarial actual exige cada vez más de sistemas informáticos capaces de adaptarse a las constantes variaciones y altos niveles de complejidad que experimentan los dominios de planificación de recursos empresariales.

Los Sistemas de Planificación de Recursos Empresariales (del inglés, ERP)[1], están destinados a esbozar el mapa de la geografía comercial que vive una empresa y que le permite anticipar caminos y obstáculos, dándole la información indispensable para la toma de decisiones. Estos sistemas, además de la gestión tradicional, brindan la facilidad de presupuestar, obtener estadísticas y generar proyecciones en base a históricos, lo que permite optimizar el rendimiento comercial. (1) (2).

Entre las características que presentan los ERP se pueden citar (3):

- El alto nivel de integración que debe existir entre cada una de las áreas de procesos que se vayan a informatizar.
- Un elevado nivel de complejidad y constantes complementaciones de diversas funcionalidades presentes en el análisis del sistema.
- Un amplio y complejo campo de negocio que se debe informatizar y muchas veces estandarizar en el dominio que se vaya a mover el sistema final.
- Un alto nivel de adaptabilidad frente los constantes cambios que sufrirá el negocio modelado. (4)

Ante este fenómeno, el competitivo mercado de desarrollo de software de gestión incrementa la presión sobre los proyectos de esta rama en materia de eficiencia y calidad de las soluciones en todos los aspectos del desarrollo: tecnológico, financiero, contractual, etc. (5)

En este sentido, la Universidad de las Ciencias Informáticas, cuenta con una línea de producción de desarrollo de productos de software para el entorno empresarial, tanto nacional como para el extranjero. El Centro de Informatización para la Gestión de Entidades (en lo adelante CEIGE), es el centro dedicado al desarrollo de proyectos de este corte dentro de la universidad, en el que se encuentra el programa ERP- Cuba.

[1]La Planificación de Recursos Empresariales, o simplemente ERP (Enterprise Resource Planning), es un conjunto de sistemas de información gerencial que permite la integración de ciertas operaciones de una empresa, (http://www.mastermagazine.info).

El programa tiene como principal objetivo el desarrollo y orquestación de una serie de sistemas que brindarán una solución nacional que permitirá, a diferencia de otros productos similares, la gestión integral de las entidades presupuestadas y empresariales, basada en los principios de independencia tecnológica y con funcionalidades generales de los procesos y las particularidades de la economía cubana. (4)

Cedrux[2], ha presentado una serie de afectaciones durante su desarrollo:

- Variación del alcance y negocio a modelar por la incorporación de nuevas funcionalidades originadas por resoluciones y decretos a nivel nacional.
- Alto nivel de dependencia e integración entre los subsistemas, que ha traído afectaciones en la calidad del producto final en los procesos de certificación y liberación a los que se ha sometido.
- El desarrollo de varios subsistemas en paralelo e integraciones con otros sistemas responsables de ciertas funcionalidades del sistema, afecta los tiempos de entregas al cliente de distintas partes del producto final.

El proceso de construcción de la arquitectura de Cedrux padece algunos de los siguientes tipos de problemas:

- Demora en el proceso de construcción de la arquitectura, atrasa próximas fases del proceso de desarrollo y conlleva a incumplimientos de cronogramas.
- Construcción de la arquitectura de forma rápida pero con insuficientes definiciones de elementos y escenarios arquitectónicos, compromete directamente la calidad del producto final.

La arquitectura de sistema define la taxonomía de empaquetamiento de los componentes abstraídos, el estilo arquitectónico a utilizar en los diferentes escenarios arquitectónicos identificados, los patrones arquitectónicos a aplicar en el diseño de los componentes, las soluciones arquitectónicas no tecnológicas orientadas a la gestión de la integración y reutilización, además de quedar definidas las soluciones orientadas a aumentar la reutilización, disminuir el acoplamiento y elevar la cohesión del diseño arquitectónico a construir. (6)

En los procesos que se llevan a cabo en el desarrollo de la arquitectura de sistema, están presentes las siguientes problemáticas:

- Poca claridad en el proceso de interpretación y análisis de los artefactos del modelado de negocio y análisis del sistema.

[2]Nombre del producto obtenido del desarrollo de los proyectos del programa ERP-Cuba, en sus distintas versiones y verticalización.

- Inconsistencias en los criterios de definición de los componentes del sistema.
- Poca especificación y documentación de las decisiones arquitectónicas tomadas.
- Pocas definiciones y estandarizaciones de los distintos elementos de integración existentes.
- Bajos niveles de reutilización en los distintos componentes de software desarrollados.
- Bajos niveles de escalabilidad en los distintos componentes de software que forman el sistema desarrollado.

Entre las causas originarias de la situación anterior, se señalan:

- Ausencia de una guía metodológica en el proceso de definición e integración de los componentes del sistema.
- Ausencia de una taxonomía de componentes, dentro de la arquitectura del sistema, ajustada a las características de los proyectos presentes en el programa ERP-Cuba.
- Ausencia de una serie de artefactos ajustados a las características del sistema a desarrollar.
- Inexistencia de una base de definiciones y especificaciones significativas para el proceso de reutilización de los distintos componentes que conforman el sistema.

Una forma de mitigar estas causas, es hacerse en cada proyecto del programa ERP-Cuba, sobre la manera de definir y reutilizar los componentes de software, lo que estime conveniente el equipo de arquitectura que lo conforme, entonces los resultados a obtener serán tan buenos como las personas del equipo sean capaces de ser.

Otra opción es contar con un modelo ajustado a las características de este tipo de software, que se le incorpore una serie de definiciones y estandarizaciones en los procesos de definición y construcción de los componentes a desarrollar, ganando con lo mismo en dominio colectivo del conocimiento y no apostar ni depender tanto de la producción heroica.

Con la primera opción los resultados serían la formación de los arquitectos de software en personas imprescindibles dentro del equipo de desarrollo de los proyectos y el éxito del proyecto sería bastante difícil de pronosticar, características estas inconciliables con los principios de producción industrial y la eficiencia de los proyectos que se necesitan.

El CEIGE, es un centro con demandas de producción industrial en el desarrollo de software y por lo tanto requiere de un sistema de producción que garantice eficiencia y resultados con una calidad homogénea. Para el logro de este objetivo es indispensable la creación de guías metodológicas, modelos y procedimientos de trabajo. (5)

Planteamiento del problema de la investigación

¿Cómo obtener los componentes de software en los proyectos del programa ERP-Cuba, de manera que se disminuyan los tiempos de desarrollo de los productos obtenidos?

Objeto de estudio

Proceso de construcción de la arquitectura de software.

Objetivo

Desarrollar una guía metodológica para la obtención de los componentes de software en los proyectos del programa ERP-Cuba, de manera que se disminuyan los tiempos de desarrollo de los productos obtenidos.

Objetivo específicos

- Establecer un marco teórico relativo a los modelos arquitectónicos utilizados para el desarrollo de la arquitectura de sistema.
- Diseñar una guía metodológica para la obtención de los componentes de software de los proyectos del programa ERP-Cuba.
- Aplicar la guía metodológica en los proyectos del programa ERP-Cuba y evaluar los resultados.

Campo de acción

La arquitectura de sistema en los proyectos del programa ERP-Cuba.

Hipótesis

El desarrollo e implantación de una guía metodológica para la obtención de los componentes de software en los proyectos del programa ERP-Cuba contribuirá a que se disminuyan los tiempos de desarrollo de los productos obtenidos.

Operacionalización de las variables dependientes e independientes

Variable independiente: guía metodológica para la obtención de los componentes de software.

Variable dependiente: los tiempos de desarrollo de los productos obtenidos.

Variable	Dimensión	Indicador
Guía metodológica	Aplicabilidad	Elementos conceptuales. (existencia)
		Guía de actividades. (existencia)

		Roles que intervienen. (existencia)
		Documentación generada (existencia)
Tiempo de desarrollo	Reutilización de componentes desarrollados.	Cantidad de componentes desarrollados.
		Cantidad de componentes reutilizados.
	Disminución de los tiempos de desarrollo.	Tiempo estimado de desarrollo.
		Tiempo real de desarrollo.
		Tiempo ahorrado.

Métodos y técnicas a utilizar

Los métodos teóricos a utilizar para estudiar las características del objeto de investigación son:

Histórico lógico: En la investigación se realiza un estudio del estado del arte con el fin de analizar la evolución que han tenido los diferentes temas asociados a la problemática (modelos, metodologías y otros elementos relacionados con la arquitectura y componentes de sistemas) y lograr un mayor entendimiento de los mismos, establecer las conexiones y dependencias fundamentales entre ellos.

Hipotético deductivo: Se sigue este método para dar solución al problema planteado, definiéndose una hipótesis a partir de la cual, siguiendo reglas lógicas de deducción, se llega a nuevos conocimientos y predicciones que posteriormente son sometidos a verificaciones empíricas.

Sistémico: El uso de este método se evidencia al tratar el problema como un todo único, estudiando la dinámica de todos los componentes que se determinan en cada uno de los elementos arquitectónicos en los proyectos, así como la relación que existe entre ellos para determinar sus dependencias e integración como sistema. Ello jugará un papel fundamental para garantizar la calidad de la guía.

Los métodos empíricos utilizados para estudiar las características de los fenómenos originados o

que involucran al objeto son:

Observación: Se hará uso de la misma durante varios momentos de la investigación, pues permite conocer de manera directa el comportamiento del objeto de estudio y como esta permite investigar el fenómeno en su manifestación externa, sin llegar a la esencia del mismo.

***Medición*:** Se sigue este método con el objetivo de obtener información numérica acerca de una propiedad o cualidad del objeto, donde se comparan magnitudes medibles y conocidas. Se hará uso de este método en los resultados que brinden las distintas validaciones a la que se sometan los resultados de la investigación.

Población y muestra

La población total está constituida por todos los proyectos del programa ERP-Cuba, para un total de diecisiete (17). 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 que por su estado dentro del proceso de desarrollo, permitiera una mejor evaluación de las variables de la presente investigación. La muestra quedó constituida por trece (13) proyectos, lo que representa un setenta y seis por ciento de la población total.

Diseño de pruebas experimentales

Para la validación de la presente investigación se hará uso de procesamiento estadísticos apoyados en los test:

- Test Mann Whitney (Comparación de dos muestras independientes).
- Test de Wilcoxon (Comparación de dos muestras apareadas)

Así como la herramienta informática SPSS para la ejecución de los test.

Para la ejecución del experimento se seguirá el siguiente diseño.

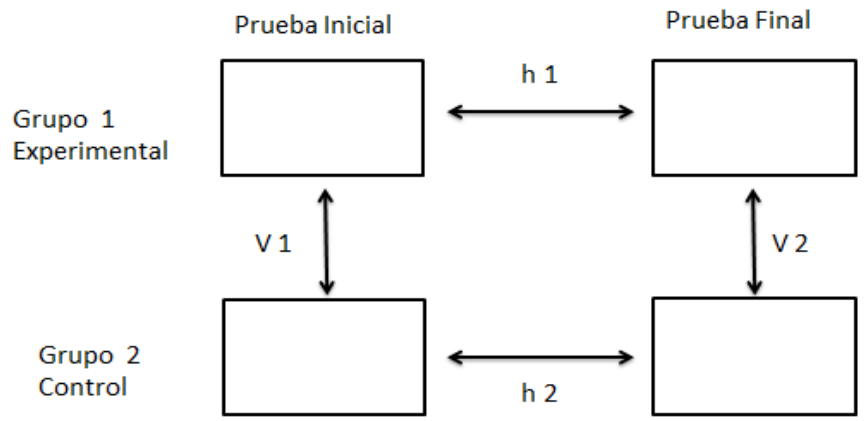


Figura 1.-Esquema General de comparación de dos poblaciones.

- Se seleccionará un grupo experimental y un grupo de control para poder realizar pruebas transversales. (v1 y v2). Dónde se aplican test para muestras independientes (Test Mann Whitney).
- Se realizarán pruebas longitudinales entre el grupo experimental antes de aplicado el experimento y luego de aplicado este para realizar las pruebas (h1 y h2). Dónde se aplican test para muestras apareadas o relacionadas. (Test de Wilcoxon).
- De la prueba v1 se espera no encontrar diferencias significativas entre las poblaciones.
- De la prueba h1 se espera encontrar diferencias significativas entre las poblaciones, demostrando que mejoró la población experimental luego de aplicado el experimento.
- De la prueba v2 se espera encontrar diferencias significativas entre las poblaciones, demostrando que mejoró significativamente la población experimental respecto al patrón.
- De la prueba h2 se realiza para tener mayor certeza de que el conjunto patrón no ha sufrido cambios significativos o al menos que los cambios significativos que ha tenido no son que ha empeorado en su desempeño.

Aporte práctico

Incidir directamente en los proyectos de desarrollo de software del programa ERP-Cuba, lográndose:

- Disminuir los tiempos de desarrollo de los proyectos.

Listado de publicaciones, eventos y avales de la investigación

- [Osmar Leyet, Larisa González][Modelo de Producción para el Flujo de Arquitectura de Sistema del Proyecto Erp-Cuba][UCIENCIA][La Habana][Cuba][19 de enero de 2010]
  [ISBN – 978-959-286-011-7].
- [Osmar Leyet, Orlando Palacio, Cesar Lage][Propuesta de empaquetamiento e integración en el sistema Cedrux] [Novena Semana Tecnológica de Fordes.][La Habana][Cuba][21 de octubre de 2009].
- [Osmar Leyet, Tahirí Rivero][Propuesta De Esquema De Producción, En El Marco De Fábrica De Software, Basado En Las Buenas Prácticas De Las Metodologías Ágiles, En El Proyecto ERP-Cuba.] [GESTEC.][La Habana][Cuba][15 de noviembre de 2010] [ISSN-1607-6281].
- [Larisa González, Osmar Leyet][Modelo de Evaluación de Arquitectura de Software del Proyecto ERP Cuba] [UCIENCIA][La Habana][Cuba][19 de enero de 2010] [ISBN - 978-959-286-011-7].

Estructura del trabajo

La presente investigación está conformada por tres capítulos. En el primer capítulo “Fundamento Teórico” se hace un análisis de los principales modelos, métodos y guías de referencias existentes en el plano internacional para la construcción de la arquitectura de sistema, así como los principales métodos de desarrollo de software basado en reutilización. Se conceptualiza y expone el estado actual de diversos conceptos arquitectónicos y se realiza una evaluación crítica de las ventajas y desventajas de diferentes enfoques y corrientes existentes sobre el tema.

En el segundo capítulo se realiza la propuesta de la guía metodológica para la obtención de los componentes de software de la arquitectura de sistema, indicando cada una de sus actividades, componentes, roles y competencias, así como los artefactos para la documentación de la vista de sistema, se conceptualiza y propone una clasificación de los componentes atendiendo a su objetivo dentro del funcionamiento del sistema.

En el tercer capítulo se realiza la validación de la presente investigación, se presenta la población y la muestra seleccionada, se presentan los indicadores de medición. Se describe el impacto de la aplicación de la guía metodológica y se presentan los resultados obtenidos de la aplicación de la misma.

CAPÍTULO 1. FUNDAMENTO TEÓRICO

1.1 Introducción

En este capítulo se especifican las diferentes escuelas o corrientes existentes en la arquitectura de software (en lo adelante, AS), se presentan los principales modelos, métodos y guías de referencias para la construcción arquitectónica de las aplicaciones actuales, así como los principales métodos existentes para el desarrollo de sistemas dirigidos por componentes. Se realiza un análisis crítico sobre estos elementos mencionados, basado en las características y demandas de los proyectos presentes en el programa ERP-Cuba.

1.2 Arquitectura de software

Philippe Krutchen, Grady Booch, Kurt Bittner y Rich Reitman definen como AS: "La arquitectura comprende un conjunto de decisiones significativas con respecto a la organización de un sistema de software incluyendo la selección de los elementos estructurales y sus interfaces que componen al sistema, el comportamiento y las especificaciones de la colaboración entre esos componentes, la interdependencia entre los elementos estructurales y de comportamiento en sistemas de gran envergadura y el estilo arquitectónico que guía esta organización" (7).

Otra definición muy referenciada y adoptada, es la que hace el Instituto de Ingenieros Electricistas y Electrónicos (del inglés, IEEE) en su estándar 1471-2000 y adoptada también por Microsoft: "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución". (8)

Basado en el estudio bibliográfico realizado y en los principios arquitectónicos a asumir en la guía metodológica que se propone, se decide adoptar la definición de AS que propone el estándar 1471-2000 de la IEEE como base conceptual de cada uno de los elementos arquitectónicos que se presentan en la investigación.

1.2.1 Modelos fundamentales

Ante el número y variedad de definiciones existentes de AS, Mary Shaw y David Garlan (9) proporcionaron una sistematización, explicando las diferencias entre definiciones en función de distintas clases de modelos. Destilando las definiciones y los puntos de vista implícitos o explícitos. Basado en los elementos presentado por estos autores, en el presente trabajo se clasifican los modelos de esta forma:

- *Modelos estructurales:* Sostienen que la AS está compuesta por componentes, conexiones

entre ellos y (usualmente) otros aspectos tales como configuración, estilo, restricciones, semántica, análisis, propiedades, racionalizaciones, requerimientos, necesidades de los participantes. El trabajo en esta área está caracterizado por el desarrollo de lenguajes de descripción arquitectónica (ADLs). (10)

- *Modelos de framework:* Son similares a la vista estructural, pero su énfasis primario radica en la (usualmente una sola) estructura coherente del sistema completo, en lugar de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA (del inglés, Common Object Request Broker Architecture), o modelos basados en CORBA, o repositorios de componentes específicos como PRISM. (10)
- *Modelos dinámicos:* Enfatizan la cualidad conductual de los sistemas. "Dinámico" puede referirse a los cambios en la configuración del sistema, o a la dinámica involucrada en el progreso de la computación, tales como valores cambiantes de datos. (10)
- *Modelos de proceso:* Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (script) de proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas. (10)
- *Modelos funcionales:* Una minoría considera la arquitectura como un conjunto de componentes funcionales organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular. (10)

Por la forma de concebir, construir y documentar una AS, se pueden identificar al menos cuatro corrientes fundamentales (9):

Arquitectura como etapa de la ingeniería de software orientada a objetos

Esta corriente arquitectónica está basada en el modelo de James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman, combinado estrechamente al mundo de UML y Rational. En esta postura, la arquitectura se restringe a las fases iniciales del proceso y concierne a los niveles más elevados de abstracción. La definición de arquitectura que se promueve en esta corriente tiene que ver con aspectos formales a la hora del desarrollo. Las definiciones revelan que la AS, en esta perspectiva, concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptibles de ser descriptas a través de las cinco vistas clásicas del modelo 4+1 de Krutchen.

Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas.

Constituye la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon en Pittsburgh: Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom. En toda la corriente, el diseño arquitectónico no sólo es el de más alto nivel de abstracción, sino que además no tiene por qué coincidir con la configuración explícita de las aplicaciones; rara vez se encontrarán referencias a lenguajes de programación o piezas de código.

Arquitectura basada en patrones

Esta corriente se basa principalmente en la redefinición de los estilos como patrones POSA (del inglés, Pattern Oriented Software Architecture), el diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura.

Arquitectura procesual

Desde comienzos del siglo XXI, con centro en el SEI (del inglés, Software Engineering Institute) y con participación de algunos, los arquitectos de Carnegie Mellon de la primera generación y muchos nombres nuevos de la segunda: Rick Kazman, Len Bass, Paul Clements, Felix Bachmann, Fabio Peruzzi, Jeromy Carriere, Mario Barbacci, Charles Weinstock, intenta establecer modelos de ciclo de vida y técnicas de diseño de requerimientos, diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la arquitectura de software.

Arquitectura basada en escenarios.

La Arquitectura basada en escenarios es la corriente más nueva. Se trata de un movimiento predominantemente europeo con centro en Holanda. Recupera el nexo de la arquitectura de software con los requerimientos y la funcionalidad del sistema, ocasionalmente borroso en la arquitectura estructural clásica. Los autores vinculados con esta modalidad han sido, los arquitectos holandeses de la universidad Técnica de Eindhoven, de la Universidad Brije, de la Universidad de Groningen y de Philips Research Mugurel Ionita, Dieter Hammer, Henk Obbink, Hans de Bruin, Hans Van Vliet, Eelke Folmer, Jilles VanGurp y Jan Bosch.

1.2.2 Consideraciones sobre las corrientes estudiadas

Las definiciones conceptuales que plantean cada una de estas corrientes, no excluyen una a las otras, ni representan un conflicto fundamental sobre lo que es o debe ser la AS.

Cada una de las corrientes mencionadas tienen exponentes en diversos modelos, métodos, metodologías y framework de desarrollo y son usadas o ajustadas para su uso en la concepción,

construcción y documentación de la arquitectura de los sistemas de software en la actualidad.

Los sistemas ERP constituyen sistemas de software de gran tamaño y altos índices de integraciones y actualizaciones en el tiempo, por lo que la construcción de la arquitectura del sistema, constituye un hito de gran significancia dentro del desarrollo del proyecto y un indicador de gran reflejo en la calidad final del sistema y el éxito del proyecto de software.

1.3 Estilos de construcción de arquitecturas

Como parte del análisis llevado a cabo en el epígrafe 1.2, donde se concluyó la existencia de diversos métodos, guías de referencias y estrategias para implementar los diversos modelos de concepción de la arquitectura de software, se pasa a realizar un estudio de las principales tendencias en el área. De un total de nueve tendencias estudiadas, se reflejan en el presente trabajo las cuatro que a opinión del autor mayores elementos presentan a tono con las características de los sistemas ERP.

1.3.1 Estrategia de arquitectura de Microsoft

La estrategia de desarrollo arquitectónico presentada por Microsoft está caracterizada por tres elementos fundamentales, el uso de un conjunto de patrones arquitectónicos que guían todo el proceso de construcción de la misma, el uso del entorno de desarrollo integrado (IDE, del inglés) como Microsoft Visual Studio Team System (11) y Microsoft Visual Studio Team Fundation Server (11), y la utilización del framework de desarrollo Microsoft Solution Framework (MSF, del inglés) (12) como guía metodológica en todo el proceso de construcción del producto.

Microsoft define cuatro vistas como ejes fundamentales en el proceso de construcción y documentación: *vista de negocios, vista de aplicación, vista de información y vista de tecnología.*

En el desarrollo de los elementos arquitectónicos de cada una de estas vistas recomienda un conjunto bastante amplio de patrones, o como le hace llamar "***Patrones & Prácticas de Microsoft".*** Estos patrones los agrupan atendiendo a los elementos arquitectónicos a los que están dirigidos (13).

Unido al conjunto de patrones reflejados en la agrupación anterior, presentan una serie de guías metodológicas del uso y aplicación de esos patrones con el uso de la Arquitectura y el marco de trabajo desarrollado por Microsoft .Net. (13)

Si bien es cierto que no se muestra abiertamente un desarrollo orientado a componentes, si es perceptible el uso de los mismos en gran parte de sus productos y sobre todo la reutilización de los mismos; y debido a la calidad y el impacto de estos en el entorno social son dignas de análisis sus

propuestas arquitectónicas.

La estrategia metodológica de Microsoft cuenta con definiciones muy precisas y aplicables a software de tipo ERP, prueba de ello es el desarrollo por dicha empresa de un sistema ERP, conocido internacionalmente como Microsoft DinamisAX. (14).

Dicha estrategia referencial está fuertemente vinculada a una serie de herramientas y definiciones muy propias de dicha empresa, que se complementan muy bien, y con la aplicación de la gran gama de patrones arquitectónicos que plantean, garantizan grandes índices de reutilización y escalabilidad de sus productos.

1.3.2 MODEL DRIVEN ARCHITECTURE (MDA)

En el año 2001, el Object Management Group (OMG) definió un marco de trabajo nuevo llamado Model Driven Architecture (MDA). La clave del MDA es la importancia que le concede a los modelos en el proceso de desarrollo de software. MDA propone la definición y uso de modelos a diferente nivel de abstracción, así como la posibilidad de la generación automática de código a partir de los modelos definidos y de las reglas de transformación entre dichos modelos. (15)

Algunas de las áreas y tecnologías que abarca, se presentan en la figura 1 que aparece a continuación:

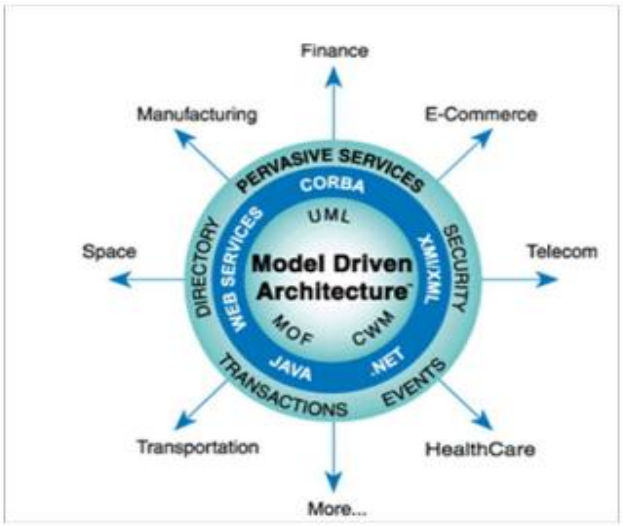


Figura 1 Áreas y tecnología del MDA.

En función del nivel de abstracción, MDA considera diferentes tipos de modelos (16):

- CIM (Modelo Independiente de la Computación)
- PIM (Modelo Independiente de la Plataforma)
- PSM (Modelo Específico de la Plataforma)
- ISM (Modelo Específico de la programación)

Para conseguir los beneficios mencionados, MDA plantea el siguiente proceso de desarrollo (15):

- Primero los requerimientos para el sistema se presentan en un modelo CIM, que describe la situación en que el sistema será usado.
- Posteriormente este modelo es transformado en un modelo PIM que describe el sistema, pero no muestra los detalles de su uso en una plataforma tecnológica particular.
- Después de obtener el modelo PIM se realiza otra transformación hacia un modelo PSM, que contiene el detalle necesario para utilizar la plataforma tecnológica en que el sistema funcionará.
- Por último teniendo el modelo PSM se realiza una transformación que resulta en la generación de código para lograr una solución o modelo ejecutable.

La idea clave de MDA es que si el desarrollo está guiado por modelos de software, se obtendrán importantes beneficios en aspectos fundamentales como (17):

- Productividad: A través de los modelos independientes de cómputo (CIM por sus siglas en inglés), independiente de plataforma (PIM por sus siglas en inglés) y de plataforma específica (PSM por sus siglas en inglés), se logran las transformaciones automáticamente, al menos en gran parte, al igual que la generación de código, permitiendo que el trabajo lo realice la herramienta y no el desarrollador.
- Portabilidad: Debido a que cuenta con un modelo PIM, todo lo definido en este modelo es portable hacia cualquier plataforma.
- Interoperabilidad: Normalmente, los modelos PSM no podrán comunicarse directamente entre ellos, ya que pueden pertenecer a distintas tecnologías. Este problema lo resuelve generando no solo los modelos PSM, sino los puentes entre ellos.
- Mantenimiento y documentación: Básicamente el modelo PIM desempeña el papel de la documentación de alto nivel que se necesita para cualquier sistema de software.

Existe una considerable cantidad de desarrollos que se están desplegando alrededor de este enfoque y diversas herramientas que lo soportan, tanto comerciales como de software libre. Estas herramientas realizan transformaciones de modelo a modelo o de modelo a código, pero ninguna de ellas toman en cuenta los mismos conceptos o características, además no consideran el proceso completo de MDA, es decir, sólo toman los modelos PIM y PSM pero no CIM. (15)

Debido a la novedad de esta corriente y la existencia de grados de inexperiencia y pruebas, no es recomendable asumir un conjunto de herramientas para la aplicación de dicha estrategia, y el desarrollo de modelos propios a partir de metamodelos existentes, sería un camino que incidiría

directamente en el tiempo de desarrollo del proyecto y sin garantía de una calidad adecuada a juzgar por las experiencias de aplicación de estos tipos de modelo.

1.3.3 Attribute-Driven Design (ADD)

Attribute-Driven Design (en lo adelante, ADD) es un método de diseño arquitectónico dirigido por los atributos de calidad que se quiere que posea el sistema, más que por la funcionalidad de la aplicación, que queda en un segundo nivel.

Las entradas para el método ADD son los requisitos de calidad del sistema expresados en forma de escenarios generales, y la salida es una arquitectura conceptual, la cual ayuda a alcanzar los atributos de calidad deseados y proporciona la estructura necesaria para dotar al sistema de la funcionalidad requerida (18) (19). Los componentes esenciales de este práctico y potente modelo de desarrollo arquitectónico son el *Plan* constituido por los atributos de calidad y las restricciones del diseño, el *Do* constituido por los elementos arquitectónicos que se instancian, para satisfacer los atributos de calidad de los requerimientos funcionales que se modelan y el *Check* constituido por el informe de análisis de que los requisitos iterados fueron cumplidos. El diseño de la arquitectura se establece mediante siete fases o pasos propuestos en el método, ver anexo 1.

El método ADD está enfocado a la definición de una arquitectura de software basado en el proceso de diseño en los atributos de calidad del sistema (20).

El método no precondiciona vistas arquitectónicas de las soluciones, aunque si usa los instrumentos de modelados. Tal vez sea este el punto débil del método, puesto que requiere una gestión arquitectónica de alto nivel y un equipo de trabajo técnicamente muy bien preparado y la inclusión de un número elevado de arquitectos de primer orden para poder dirigir técnicamente el proceso, pero ¿se podría asegurar que siempre se contará con un equipo de arquitectos altamente preparados y con suficiente experiencia en el trabajo mediante esta metodología?, es muy difícil responder positivamente la pregunta anterior, ahora, debido a la especificidad, validez, y ajustes de los procesos de desarrollo de la arquitectura que propone la guía del SEI, con las características de las necesidades existentes anteriormente explicadas, el método constituye, sin lugar a dudas, referencia sólida en el desarrollo de la presente investigación.

1.3.4 Arquitecturas orientadas a componentes

Otro enfoque muy utilizado en la actualidad para el desarrollo de arquitecturas lo constituye las arquitecturas orientas a componentes.

El disponer de componentes software no es suficiente para desarrollar aplicaciones, ya provengan estos de un mercado global o sean desarrollados a medida para la aplicación. Un aspecto crítico a la hora de construir sistemas complejos es el diseño de la estructura del sistema. (7)

Visto así, para explotar las bondades de un desarrollo orientado a componentes, es necesario concebir incluso desde la columna central del sistema, la arquitectura, orientada a componentes.

En general, dicha representación se va a realizar en términos de una colección de componentes y de las interacciones que tienen lugar entre ellos.

De esta forma aparecen las arquitecturas basadas en componentes y conectores: los primeros se dedican a labores computacionales, mientras que los conectores encapsulan los patrones de sincronización y coordinación entre los componentes.

Estos tipos de arquitecturas son completamente modulares y favorecen la reutilización de todos sus elementos, incluyendo los que definen las distintas relaciones entre ellos (21).

Existe un número de bondades que ofrece el uso de componentes de software como concepto fundamental en la construcción de una arquitectura, dos de esas principales bondades la constituyen (21):

Reutilización: Habilidad de un componente de software de ser utilizado en contextos distintos para los que fue diseñado (reutilizar no significa usar más de una vez).

Existen 4 modalidades de reutilización, dependiendo de la cantidad de información y posibilidades de cambio que permita el componente a ser reutilizado: caja blanca, caja de cristal, caja gris y caja negra.

Polimorfismo: Habilidad de un mismo componente de mostrarse de diferentes formas, dependiendo del contexto; o bien la capacidad de distintos componentes de mostrar un mismo comportamiento en un contexto dado. Ambas acepciones representan los dos lados de una misma moneda.

Se han propuesto diversos modelos de arquitecturas software usando componentes y conectores que se basan en la especificación de la arquitectura del sistema. Todos ellos comparten la misma visión arquitectónica de los sistemas, aunque cada uno de ellos utilice notaciones y formalismos diferentes.

1.3.5 Consideraciones sobre los estilos estudiados

La serie de modelos, guías y métodos estudiados hasta ahora exponen desde distintos estilos de qué forma realizar este conjunto de acciones. Cada uno de estos modelos conceptualizan parte de

las actividades fundamentales y se apoyan o destacan determinados elementos sobre otros dependiendo fundamentalmente del tipo de sistema para el cual se esté realizando el proceso de construcción arquitectónica.

Basado en las definiciones y temas abordados referentes al desarrollo orientado a componentes, para el desarrollo de la arquitectura de sistema, se decide hacer uso de dicha corriente para la construcción de la arquitectura de los proyectos del programa ERP-Cuba. Haciéndose necesario contar con un procedimiento para la obtención de los componentes del sistema, así como una taxonomía de los componentes a definir.

1.4 Desarrollo de software basado en componentes

El desarrollo de una arquitectura orienta a componentes, conclusión de 1.3.5, basa su estructura fundamental en el concepto componente, las interacciones entre estos y la reutilización como herramienta fundamental para el desarrollo de la misma y del sistema en cuestión.

1.4.1 Componentes

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 (22).

- 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 (23).
- Por su parte Martínez (22) 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.
- Según Philippe Krutchen de Rational Rose (24), un componente es una parte no trivial, casi independiente y reemplazable de un sistema que cumple una función dentro del contexto de una arquitectura bien definida. Un componente cumple con un conjunto de interfaces y provee la realización física de ellas.

En esta investigación se adopta como definición genérica de componente la de Clemens Szyperski: Unidad de composición con interfaces especificadas contractualmente y solamente dependencias explícitas de contexto.

Un componente de software puede ser desplegado independientemente y está sujeto a composición por terceros (25). Con el posterior desarrollo de la presente investigación será

necesario hacer una acotación más específica de este concepto, según las características del sistema y el desarrollo arquitectónico a establecer.

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. A continuación se presentan algunas características de los componentes, que complementan esta definición (26):

- *Identificable*: 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.

Adicionalmente para favorecer su reutilización, es deseable que un componente sea (27):

- *Genérico:* sus servicios pueden ser usados en una gran variedad de aplicaciones.
- *Autocontenido:* es conveniente que un componente dependa lo menos posible de otros componentes para cumplir su función de forma tal que pueda ser desarrollado, probado, optimizado, utilizado, entendido y modificado individualmente.
- *Mantenido:* es deseable que un componente (como toda pieza de software) esté inmerso en un proceso de mejoramiento continuo que le garantice al integrador nuevas versiones que incluyan correctivos, optimizaciones y nuevas características. Esto contribuye a que dicho componente sea seleccionado con mayor frecuencia para formar parte de sistemas de software.
- *Certificado:* el componente puede ser certificado por una agencia de software independiente o mediante la aplicación de modelos de auto-certificación que le permiten al comprador del componente determinar la calidad del software adquirido.

1.4.2 Interacción entre componentes

En el desarrollo de sistemas orientados a componentes resulta de vital importancia la definición y control de las interacciones a producirse entre los distintos componentes que forman el sistema.

La integración entre componentes es definido por Duvall (28) 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, que pueden llegar a ser sólo archivos de código de bajo nivel compilados para pequeños proyectos.

Uno de los conceptos de gran valor e importancia en el proceso de interacción es la interfaz que presentará cada uno de los componentes definidos.

Una interfaz define el conjunto de operaciones que un componente puede realizar; estas operaciones son llamadas también servicios o responsabilidades. Las interfaces proveen un mecanismo para interconectar componentes y controlar las dependencias entre ellos. En muchas ocasiones se suele declarar la interfaz del componente en el mismo lenguaje de programación usado para implementar el resto de las funcionalidades, atentando contra la característica de la generisidad que deben presentar los mismos. (26)

En la búsqueda de eliminar las posibles afectaciones por dicha práctica, surgen algunos lenguajes neutrales de especificación de interfaces, tales como IDL (Interface Definition Language) (29) de OMG (Object Management Group). En general, una interfaz de programación de aplicaciones (API, Application Programming Interface) es una especificación, en un lenguaje de programación, de las propiedades de un módulo de software. Los clientes del módulo sólo deben depender exclusivamente de las propiedades definidas por el API de forma explícita.

Por otra parte, las interfaces convencionales definen la firma de las operaciones (nombre de la operación, tipo y orden de los argumentos, y la manera como se devuelven los resultados) que provee un componente (26).

Las operaciones también se conocen como propiedades funcionales. Sin embargo, estas interfaces no expresan adecuadamente propiedades del componente relativas a, por ejemplo, su desempeño, precisión, disponibilidad, latencia, seguridad, entre otras. Dichas propiedades se conocen como propiedades extra funcionales (30).

Además de los componentes, los framework también se consideran entidades sujetas a composición. En consecuencia, existen tres clases principales de interacción en los sistemas basados en componentes (30):

- Componente-Componente (C-C): permite la interacción entre componentes. De este tipo de interacción se obtiene la funcionalidad de la aplicación, de forma tal que los contratos que especifican este tipo de interacción pueden ser clasificados como Contratos a Nivel de Aplicación.
- Componente-Framework (C-F): posibilita las interacciones entre el framework y sus componentes. Dicha interacción permite que el framework administre los recursos de los

componentes. Los contratos que especifican estas interacciones pueden ser clasificados como Contratos a Nivel de Sistema.

- Framework-Framework (F-F): posibilita las interacciones entre framework y permiten la composición de componentes desplegados en framework heterogéneos. Estos contratos pueden ser clasificados como Contratos de Interoperabilidad.

La forma de materializar la composición entre componentes depende de los mecanismos especificados por su modelo de programación y depende mucho de la taxonomía de componentes que se defina a obtener en el dominio en cuestión, así como la estructura de empaquetamiento y los tipos de componentes por nivel que presentará el sistema.

El uso de componentes como guía en el proceso de desarrollo de un proyecto y columna central del sistema tiene una incidencia directa en la arquitectura de software, provee a la misma la reafirmación o el fomento de un conjunto de atributos de calidad, que tienen una incidencia directa en la calidad final del sistema.

Otro elemento de gran significado es el impacto directo en los tiempos de desarrollo mediante el fomento de la reutilización de componentes.

1.4.3 Reutilización de componentes

La reutilización de componentes de software es un proceso inspirado en la manera en que se producen y ensamblan componentes en la ingeniería de sistemas físicos.

Existen variadas definiciones del término Reutilización de Software. Algunas de estas definiciones son las siguientes:

• Según Somerville (31), la reutilización es un enfoque de desarrollo [de software] que trata de maximizar el uso recurrente de componentes de software existentes.

• Según Sametinger (32), la reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo.

• Según Sodhi (33), la reutilización de software es el proceso de implementar o actualizar sistemas de software usando activos de software existentes.

Estas tres definiciones consideran la reutilización como un enfoque o proceso de desarrollo de software. Su principal diferencia radica en el objeto de reutilización (componente, software y activo).

Varios estudios han demostrado la efectividad de la reutilización del software. Sametinger (32), en particular, describe algunos de estos indicadores:

- Entre el 40 y 60% del código fuente de una aplicación es reutilizable en otra similar.
- Aproximadamente el 60% del diseño y del código de aplicaciones administrativas es reutilizable.
- Aproximadamente el 75% de las funciones son comunes a más de un programa.
- Sólo el 15% del código encontrado en muchos sistemas es único y novedoso a una aplicación específica.
- El rango general de uso recurrente potencial está entre el 15% y el 85%.

A partir de estos indicadores es fácil deducir el impacto y la importancia que tiene la reutilización de componentes en el proceso de desarrollo de software; particularmente, en tres de las variables más importantes de la Ingeniería de Software: el costo, tiempo y esfuerzo requerido para desarrollar un producto de software (34). La aplicación apropiada de la reutilización en un proyecto de software conduce, indiscutiblemente, a una reducción significativa de los valores de estas tres variables. Otros beneficios importantes son el incremento de la calidad del software producido, el aumento de la productividad de los grupos de desarrollo y la reducción del riesgo global del proyecto. (35)

En el contexto de Ingeniería de Software, un Activo Reutilizable es un producto diseñado expresamente para ser empleado de forma recurrente en el desarrollo de muchos sistemas y aplicaciones. Ejemplos de activos reutilizables son: algoritmos, patrones de diseño, esquemas de base de datos, arquitecturas de software, especificaciones de requisitos, de diseño y de prueba, componentes entre otros. (35)

1.4.4 Consideraciones sobre los conceptos estudiados

En el epígrafe 1.3.5, se concluyó el uso del estilo orientado a componentes para el desarrollo de la arquitectura de los proyectos del programa ERP-Cuba, basado en las facilidades y oportunidades que ofrece este estilo y las características de los sistemas ERP.

Para la aplicación de un estilo orientado a componentes es necesario la definición de la taxonomía de componentes a utilizar de acuerdo a las características de los proyectos, la especificación de las formas de interacción o integración a efectuarse entre los componentes definidos y otro elemento de suma importancia son los procedimientos para obtener los componentes del sistema y propiciar luego la reutilización.

1.5 Métodos para la obtención de componentes

El desarrollo de sistemas haciendo uso de componentes de software es un método bastante utilizado en la actualidad, sus ventajas y características han propiciado la aparición de una serie de

modelos y métodos que precisamente sirven de base al proceso de desarrollo.

Durante el proceso de investigación se revisaron siete modelos relacionado con el desarrollo de sistemas basado en reutilización, atendiendo a su representatividad se decidió profundizar en tres de ellos, el método WATCH creado específicamente para el desarrollo y reutilización orientada a componentes y su variante WATCH – COMPONENT, el método FODA, propuesto por el SEI, basado en la experiencia , jerarquía mundial e integraciones con otra serie de modelos arquitectónicos propuestos igualmente por esta organización , el método JODA, uno de los más referenciados en el plano internacional en cuanto al análisis de dominios.

1.5.1 WATCH-COMPONENT

Modelo para la obtención de componentes para reutilización. Modelo propuesto en la Universidad de Los Andes (Venezuela) para el desarrollo de aplicaciones empresariales. (36)

Es un modelo desarrollado expresamente para producir componentes de software reutilizable.

Consta de tres modelos específicos (37):

- Modelo del producto: captura las propiedades de los componentes de software reutilizables.
- Modelo del proceso: describe las actividades necesarias para producir un componente de software reutilizable.
- Modelo del grupo de desarrollo: describe los actores y roles del grupo de desarrollo de los componentes de software reutilizables.

En el modelo del producto se hace una conceptualización de los principales elemento que se abordaran en el modelo, se hace referencia a una serie de artefactos y procesos desarrollados en fases de la Ingeniería de dominio, que si bien es cierto el presente método se enmarca solo en la ingeniería de componentes, si tiene una fuerte integración y dependencia de la forma de realizar la ingeniería de dominio dentro del proceso de desarrollo (38).

En el modelo de proceso se describe cada una de las actividades por la que pasa la construcción del componente, proponiendo una forma cíclica del proceso y con el mismo el ciclo de vida del componente (38).

El modelo del grupo de desarrollo recoge la estructura organizativa y los roles que intervienen en los proyectos donde se desarrollan cada uno de los componentes que guiarán todo el proceso de desarrollo (38), ver figura 2.

Figura 2. Estructura organizativa para el desarrollo de software que propone Watch.

Watch maneja dos modelados principales desde el punto de vista conceptual:

- El modelado de los procesos de desarrollo, que tiene la finalidad de mostrar las etapas que deben ejecutarse para el desarrollo de aplicaciones. Cada una de estas fases se describe hasta llegar al nivel de tareas, se describen los productos y se propone una técnica, describiéndose los actores y roles correspondiente en cada fase.
- Modelado de los procesos gerenciales. Procesos escogidos de acuerdo a los estándares IEEE 1074 y SIPCE (ISO 15504) que corresponden a estándares de ciclos de vida de software.

Para el desarrollo de la presente investigación se realiza un estudio más detallado referente a los procesos de desarrollo.

El proceso de desarrollo está compuesto por ocho fases:

- Modelado de Negocio.
- Definición y Especificación de requerimientos.
- Diseño de la aplicación.
- Especificación de componentes.
- Implementación de componentes.
- Ensamblaje de componentes.
- Prueba de aplicación.
- Entrega de la aplicación.

De cada una de estas fases se especifica el flujo de actividades correspondiente, los artefactos a generar como parte de la documentación y los roles involucrados con sus correspondientes responsabilidades.

1.5.2 FODA

La metodología FODA (Características orientadas al análisis de dominios, del inglés Feature-Oriented-Domain-Analysis) fue desarrollada por el SEI (Software Engineering Institute) de la universidad de la Carnegie Mellon.

Soporta el descubrimiento, el análisis y la documentación de los aspectos comunes y las diferencias en un dominio. El descubrimiento sistemático y explotación de aspectos comunes es una técnica fundamental para encontrar los requerimientos y lograr satisfactoriamente la reutilización de software en sentido general, pudiéndose establecer los productos a desarrollar y reutilizar como componentes del sistema. (39)

El método de análisis FODA propone mediante el análisis de dominio obtener una descripción genérica de los requisitos y una forma de implementarlos. El resultado de aplicar FODA es una familia de productos más que un producto individual, produciendo un modelo del dominio con la flexibilidad suficiente para reflejar las diferentes posibilidades que existan, y una arquitectura estándar para desarrollar componentes software. En la figura se muestran todas las fases y productos del análisis del dominio propuesto por FODA (38).

Tabla 1. Fases del método FODA.

FASE	ENTRADAS	PROCESOS	PRODUCTO	DESCRIPCIÓN
Análisis del Contexto	Estándares, ambientes operacionales.	Análisis del Contexto.	Modelo del Contexto.	Ambiente en los cuales las aplicaciones pueden ser utilizadas u operadas.
Modelado del Dominio	Características. Modelo del contexto.	Análisis de las características.	Modelo de características.	Perspectivas de los usuarios finales de las capacidades de una aplicación en un dominio.
	Aplicación del conocimiento del dominio.	Modelado entidad relación.	Modelado entidad relación.	Los desarrolladores entienden las relaciones de las entidades dentro del dominio.

	Tecnología del dominio, Modelo del contexto, Modelo de características, Modelo entidad relación, requerimientos.	Análisis funcional.	Modelo de flujo de datos. Modelo de máquinas de estado finito.	Perspectiva de requerimientos de los analistas de la funcionalidad de las aplicaciones.
Modelado de la Arquitectura	Implementación tecnología, Modelo de contexto, Modelo de características, Modelo entidad relación, Diseño de la información.	Modelo de la arquitectura.	Modelo de procesos de interacción, Diagrama de estructura de los módulos.	Perspectiva de los diseñadores de la arquitectura de la aplicación.

En la tabla 1 se muestra el modelo de procesos del método FODA, descrito a través de sus fases, entradas, procesos y productos.

FODA organiza las necesidades de los usuarios en jerarquías en forma de árbol. Las capacidades o características comunes se colocan en los nodos raíz del árbol. Cada variante sobre estas características comunes implica una rama en el árbol, ver figura 3.

La ventaja de esta técnica es que se recogen en un único modelo todas las posibles variaciones que hay en la familia de productos y que se identifican claramente. Las capacidades pueden ser obligatorias, opcionales o alternativas (40).

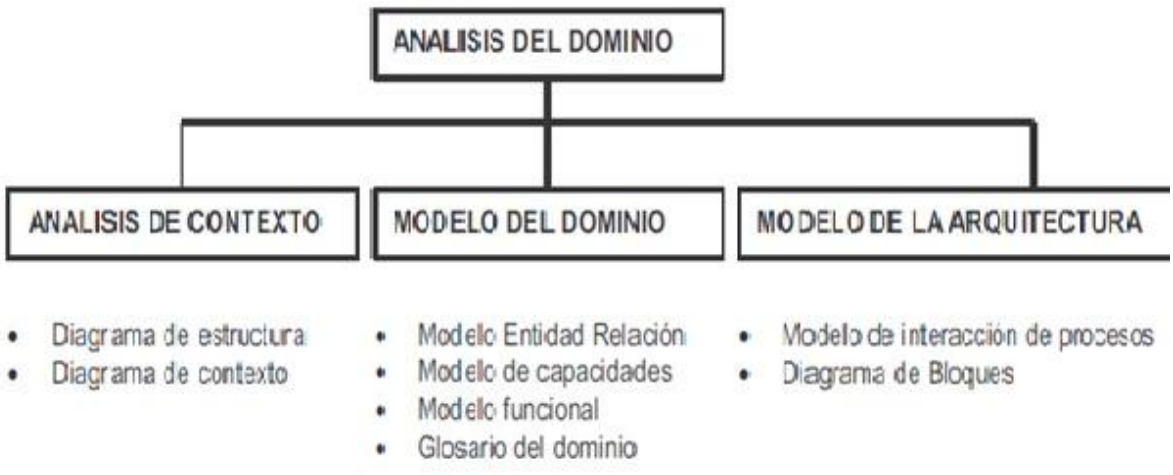


Figura 3. Mapa conceptual del funcionamiento de FODA.

El método FODA, si bien es cierto que en su versión original no tiene una clara orientación al concepto componente, se hace evidente que impera dentro de sus bases la reutilización de activos,

idea bastante cerca del desarrollo basado en componentes.

1.5.3 JODA

Joint Integrated Avionics Working Group (JIAWG) Oriented Domain Analysis (JODA) es un método que es utilizado por el JIAWG para el análisis de dominio. Este método está basado en el análisis orientado a objetos y la notación definida por Coad and Yourdon, las cuales son referidas como Coad Yourdon Object Oriented Análisis (CYOOA) (41).

Este método adiciona ciertas extensiones ya que la notación original está hecha para la construcción de un único sistema. CYOOA fue mejorada para validar y demostrar la efectividad de técnicas orientadas a objetos para soportar el Análisis de Dominio (38).

Joda incide directamente en la ingeniería de dominio teniendo como principal salida, un modelo de domino orientado a la reutilización de objetos como activo principal. Para el logro de su objetivo propone tres procesos fundamentales secuenciales que tienen como entrada cada uno la salida de su proceso predecesor, tal como se muestra en la siguiente figura 4.

ETAPA	ENTRADAS	PROCESO	SALIDAS	DESCRIPCIÓN
Análisis de Dominio	Material fuente, expertos, información de sistemas anteriores.	Preparación del Dominio	Material fuente relevante	Identifica y colecciona fuentes relacionadas con la definición del Dominio.
	Material Fuente Relevante	Definición del dominio.	Definición preliminar del dominio. Que incluye: Diagramas de alto nivel Diagrama del Todo Diagrama generalización especialización. Dominios de Servicio Dependencias de los dominios Glosario del dominio Descripción textual	Limitar el dominio e identificar que se puede y que no reutilizar.
	Definición preliminar del dominio	Modelado del Dominio. Examinar la historia de los objetos. Abstraer y agrupar objetos	Definición del Dominio y Modelo del Dominio Diagrama de Clases Diagrama de Estructura Evolución del modelo del dominio.	El modelo del dominio que se va a utilizar para la reutilización es depositado en un repositorio,

Figura 4. Procesos del método JODA.

Joda hace un especial énfasis en el modelado del dominio y en la preparación de las bases, traducido en una serie de artefactos para la reutilización de los activos, nombrados específicamente como objetos en este método. Incorpora de manera directa el uso de un repositorio para el activo reutilizable, pero deja de manera poco clara, la especificación y características del activo, así como las formas de interacción entre los mismos y no llega a niveles

de proponer una clasificación o configuración de los mismos, concepto este necesario para el uso de componentes de software, como elemento clave para la arquitectura de sistema y proceso de desarrollo del proyecto.

1.5.4 Consideraciones sobre los métodos estudiados

De los modelos estudiados se considera Watch como el más abarcador e integral, ya que tanto FODA como JODA se concentran solo en la ingeniería de dominio, dejando a un segundo plano los importantes procesos de la ingeniería de aplicación (procesos de desarrollo de componentes y soluciones a partir de componentes existentes), siendo estos procesos de vital importancia en los proyectos del programa ERP-Cuba, que constantemente construyen soluciones a la medida para escenarios específicos de despliegue, sobre la base de los componentes desarrollados como parte de las soluciones genéricas que conforman el sistema Cedrux.

El método Watch por su parte en el proceso número cuatro, referente a la especificación de componentes es donde enmarca las actividades relacionadas con el proceso de obtener los componentes del sistema, del análisis de estas actividades se puede concluir:

- En el modelo de proceso, Watch propone la organización del equipo para el desarrollo del proyecto dirigido por componentes, por las características de los proyectos del programa ERP-Cuba, en los cuales se desarrollan en forma paralela procesos de formación a estudiantes del segundo ciclo, procesos de investigación de la actividad posgraduada y un conjunto de grandes compromisos productivos, no es conveniente adoptar la estructura que se propone en el modelo.
- Claro enfoque y orientación a componentes que presentan interfaces de cara al cliente sobre la web, condición esta que no aplica a todos los proyectos del programa ERP-Cuba.
- No se tienen en cuenta los atributos de calidad a favorecer en el proceso de definición de los componentes, ni se hace una valoración de la relación decisiones del diseño – atributos de calidad.
- Al no especificarse una taxonomía de componentes, teniendo en cuenta diversos tipos de conceptos presentes en las descripción de la solución a implementar, es imposible aplicar la meta arquitectura que propone Watch a todos los componentes a definir en los proyectos del programar ERP-Cuba.

1.6 Conclusiones Parciales

Se concluye que:

- El contar con una arquitectura orientada a componentes le ofrece al sistema final un conjunto

de indicadores con resultados positivos en la calidad final del producto. Por tanto es una decisión el asumir la construcción de la arquitectura base de los proyectos orientada a componentes de sistema y por consiguiente contar con un marco metodológico o de referencia, que conceptualice la taxonomía de componentes a utilizar, la configuración y documentación de estos, así como la forma de obtener e integrar los componentes.

- La revisión de los principales métodos, modelos, guías y estrategias para la construcción de la arquitectura y para específicamente la obtención de componentes de sistema, demostró la existencia de dificultades en la adecuación de estos para su aplicación en los proyectos del programa ERP-Cuba por diversos factores explicados en 1.5.4.
- Los elementos analizados en las secciones anteriores apuntan la necesidad de definición y especificación de una guía metodológica que defina la taxonomía de componentes necesaria a obtener, configurar, integrar y documentar en el contexto y características de los proyectos del programa ERP-Cuba.

CAPÍTULO 2. PROPUESTA METODOLÓGICA

2.1 Introducción

En el presente capítulo se darán a conocer los elementos que conforman la propuesta metodológica, sus características, implicaciones y restricciones. Se abordan los elementos distintivos del mismo, los pasos metodológicos para su aplicación, la conceptualización de cada uno de los elementos que se utilizan, así como los roles involucrados en la puesta en práctica de la propuesta.

2.2 Elementos metodológicos de la propuesta

La metodología que se utilizará para el desarrollo de este trabajo está fundamentada en la ingeniería de métodos. Un método en la ingeniería de métodos (42), corresponde al conjunto de definiciones de un procedimiento más un modelo que permita construir un sistema de software y está compuesto por:

- El Modelo de Producto, representa el conjunto de conceptos que se pueden utilizar para construir un producto o sistema. (¿Qué?)
- El Modelo de Proceso, permite construir el producto o sistema dando directivas y/o prescripciones que guían las actividades y tareas de desarrollo del producto. (¿Cómo?)
- En la última versión del Modelo WATCH (26) se añade a este método un Modelo del Grupo, que corresponde a los roles jugados por los participantes del proceso de desarrollo. (¿Quiénes?)

La ingeniería de métodos consiste en aplicar enfoques, técnicas y herramientas de ingeniería a la construcción de métodos. Es una disciplina de diseño, construcción y adopción de métodos y herramientas para el desarrollo de sistemas de software. Es un enfoque coordinado y sistemático para establecer métodos de trabajo, ver figura 5.

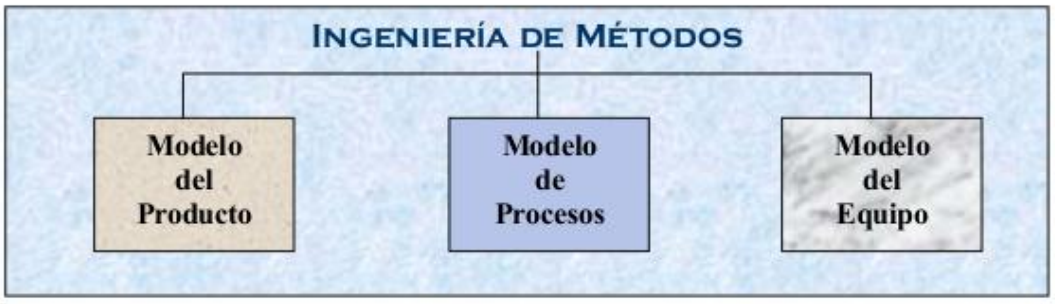


Figura 5. Modelos de la Ingeniería de métodos.

2.3 Modelo del producto

2.3.1 Incidencia en las vistas arquitectónicas

La arquitectura de sistema define la taxonomía de empaquetamiento de los componentes abstraídos, el estilo arquitectónico a utilizar en los diferentes escenarios arquitectónicos identificados, los patrones arquitectónicos a aplicar en el diseño de los componentes, las soluciones arquitectónicas no tecnológicas orientadas a la gestión de la integración y reutilización, además de quedar definidas las soluciones orientadas a aumentar la reutilización, disminuir el acoplamiento y elevar la cohesión del diseño arquitectónico a construir. (43)

Los componentes de software contendrán la implementación de las funcionalidades del sistema, así como las validaciones y configuraciones establecidas por el negocio. Por tales motivos la presente propuesta metodológica incide directamente en la vista de componentes y la vista de integración, de las vistas arquitectónicas definidas para los proyectos del programa ERP-Cuba (44). En el contexto de los proyectos del programa ERP-Cuba, estas vistas brindarán respectivamente cada uno de los siguientes elementos:

Vista de Integración: Encargada de los procesos de integración a nivel de sistema.

- Integración interna (entre componentes de un mismo subsistema).
- Integración externa (entre distintos subsistemas).
- Interoperabilidad entre distintos sistemas con necesidad de establecer comunicación.
- Establece las definiciones, estándares, restricciones, protocolos de comunicación y reglas de intercambio de información.

Vista de Componentes: Encargada de las definiciones de la taxonomía de componentes.

- Especificación de la composición estructural interna de cada uno de estos componentes (vista vertical de la arquitectura).
- Definir la estructura de empaquetamiento del sistema.
- Definición de cada uno de los componentes que conforman el sistema atendiendo a la estructura de empaquetamiento a utilizar.
- Definición de cada uno de los componentes que conforman el sistema atendiendo a la función dentro del sistema.
- Seguimiento y control del ciclo de vida asociado a cada uno de los componentes del sistema.

2.3.2 Taxonomía de componentes

La vista de componentes es una de las subdivisiones que se encuentran en la vista de sistema (44),

es la encargada de las definiciones de los tipos de componentes posibles a definir en el proyecto, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes (vista vertical de la arquitectura).

En el marco del presente trabajo, se amplía el concepto de componente asumido en 1.4.1, estableciendo como concepto el siguiente: *Constituye una abstracción de parte o la totalidad de determinadas áreas funcionales, reglas del negocio o del sistema, definiciones tecnológicas implementadas o flujos definidos como parte del proceso de configuración del sistema, es la menor unidad existente en la estructura de empaquetamiento del sistema, autónomo en su funcionamiento y capaz de implementar interfaces o contratos de comunicación.*

Atendiendo a la estructura de empaquetamiento se proponen dos tipos de componentes:

Primer nivel ó nivel 1 -> Componente: Definido anteriormente.

Segundo nivel ó nivel 2 -> Componente contenedor: Estructura formada por un conjunto de componentes (nivel 1) que interactúan entre ellos, cuya agrupación se lleva a cabo para dar respuesta a un área funcional o parte de la misma, que como área tiene razón de existencia, independencia funcional e interacción con otros elementos, ver figura 6.

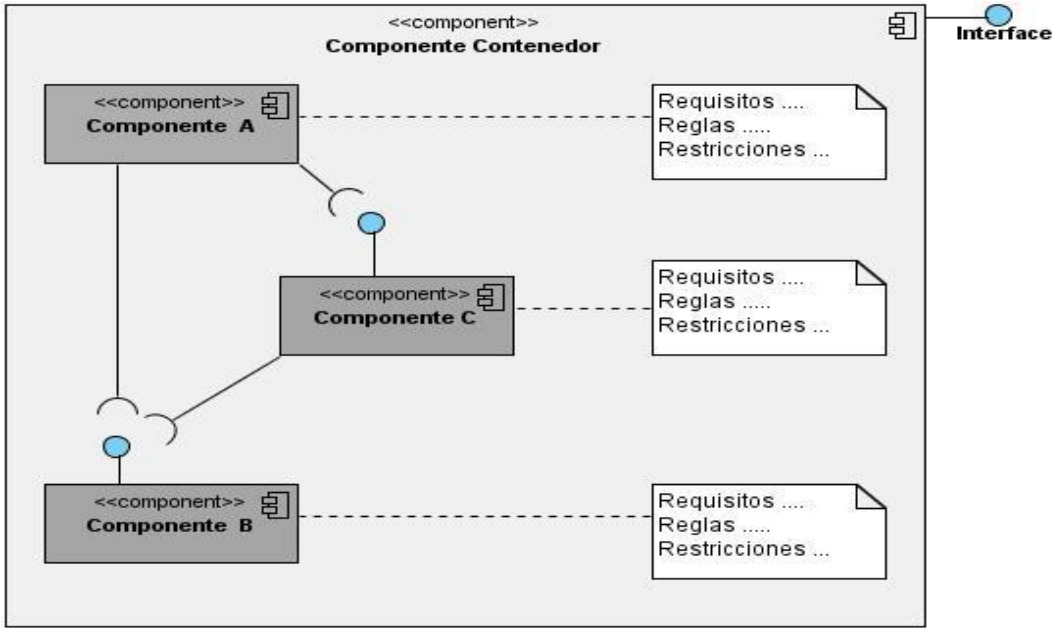


Figura 6. Componente contendor de la taxonomía propuesta.

De esta forma la incidencia en la vista de sistema de la arquitectura de software quedaría conformada por la interacción del conjunto de componentes contenedores y la interacción en el interior de cada componente contenedor llevado a cabo por los componentes del primer nivel que lo conforman, más la metodología para la obtención de estos componentes.

Los componentes de primer nivel guiarán el desarrollo del sistema e implementarán la trazabilidad directa de las funcionalidades del sistema expresada en requisitos funcionales, reglas del negocio, restricciones del diseño y las decisiones arquitectónicas tomadas durante todo el proceso de diseño y concepción del sistema.

Por su parte los componentes contenedores se obtendrán como parte del proceso de empaquetamiento y ajuste de una instancia del sistema, para la obtención de una solución específica.

Según la función dentro del sistema se proponen cinco tipos de componentes:

Categoría Núcleo

Representan aquellos componentes con la responsabilidad de abstraer las principales funcionalidades del negocio de la organización, comúnmente agrupan la mayor parte de los conceptos del modelo conceptual, y en términos arquitectónicos agrupan el deseo funcional del sistema que solicita el cliente.

Categoría Configuración

Representan aquellos componentes con la responsabilidad de abstraer las configuraciones y parametrizaciones estáticas o dinámicas del sistema, así como las características de los estándares a utilizar en la conversión de formato y validaciones de procesos del negocio.

Categoría Integración

Representan aquellos componentes con la responsabilidad de abstraer la estrategia de integración y colaboración del sistema, así como los flujos y subflujos de trabajo. Otra de las responsabilidades arquitectónicas que descansan en este grupo de componentes o elementos arquitectónicos es la interoperabilidad e intraoperabilidad con otras plataformas tecnológicas afines al sistema en cuestión o de otro tipo.

Categoría Tecnología

Representan aquellos componentes con la responsabilidad de abstraer características puramente tecnológicas que sirven de base tecnológica al resto de los componentes del sistema. Tienen la responsabilidad de contribuir directamente con la abstracción de las características no funcionales, con el objetivo de lograr una separación arquitectónica y una mejor especialización entre los diferentes grupos de arquitectos de soluciones.

Categoría Prueba

Representan aquellos componentes con la responsabilidad de abstraer las pruebas unitarias y de integración del resto de los componentes de los otros grupos. Por el costo y el esfuerzo que requieren generalmente son asociados solamente a los componentes núcleos, con el objetivo de poder maquetar la solución y garantizar que el núcleo funcional es válido, sin embargo puede vincularse a cualquiera de los restantes tipos de componente.

Todos los componentes clasificados en una de estas cinco categorías, atendiendo a su función dentro de la arquitectura del sistema, pertenecen al tipo de componente de primer nivel, según la clasificación atendiendo a la estructura de empaquetamiento, ver anexo 2.

2.3.3 Interacciones entre componentes

La interacción entre componentes, centra, orienta y valida la comunicación existente en el sistema, así como la posible interacción del sistema con sistemas o fuentes de informaciones externas. En un modelo de desarrollo de software orientado a componentes, como el que se adopta en el trabajo presente, donde los equipos de desarrollo se centran en la producción de componentes y no de sistemas, el proceso de ensamblado de estos componentes a conveniencia para lograr diversos tipos de sistemas, según los intereses o compromisos en esos momentos puede resultar tan eficiente y productivo como interminable y caótico, la diferencia lo hace el proceso de estandarización y preparación de cada una de las posibles integraciones a producirse en el sistema.

Por la disposición de la configuración del sistema, se identifican dos tipos de interacciones entre los componentes:

Una primera interacción es la que se establece entre los componentes de primer nivel en el interior de un componente contenedor. En la interacción entre el componente que consume el servicio y el componente que lo brinda, existen tres elementos fundamentales:

- La especificación del servicio brindado (Contrato).
- El motor de integración.
- La especificación del consumo del servicio (Subscripción).

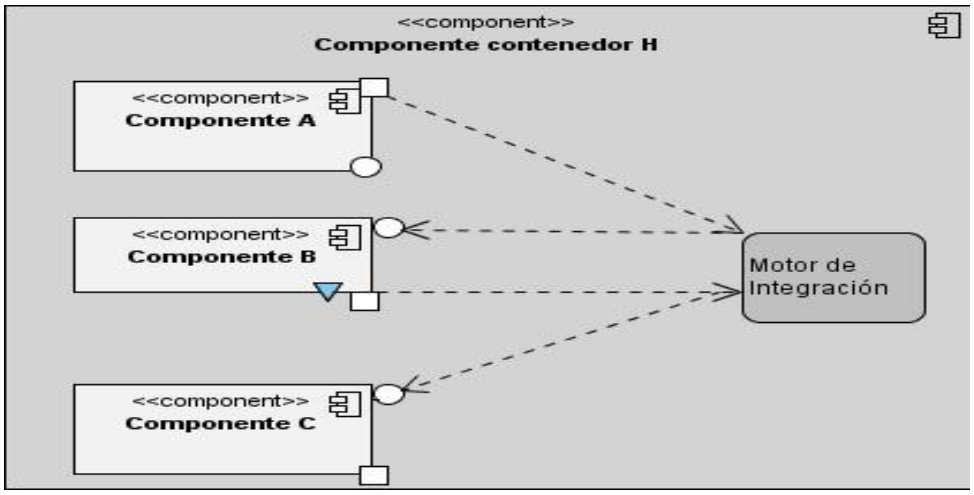


Figura 7. Interacción entre componentes nivel 1.

Cada componente contiene una interfaz donde publica los servicios que brinda, y otro paquete (puerto de salida), donde se subscribe y realiza el llamado al motor de integración de los servicios a consumir, de manera que no existe una dependencia directa entre dos componentes, sin ser controlado por el motor de integración, ver figura 7.

El segundo nivel de interacción es el que se establece entre los componentes contenedores del sistema y tiene una gran semejanza con el explicado anteriormente:

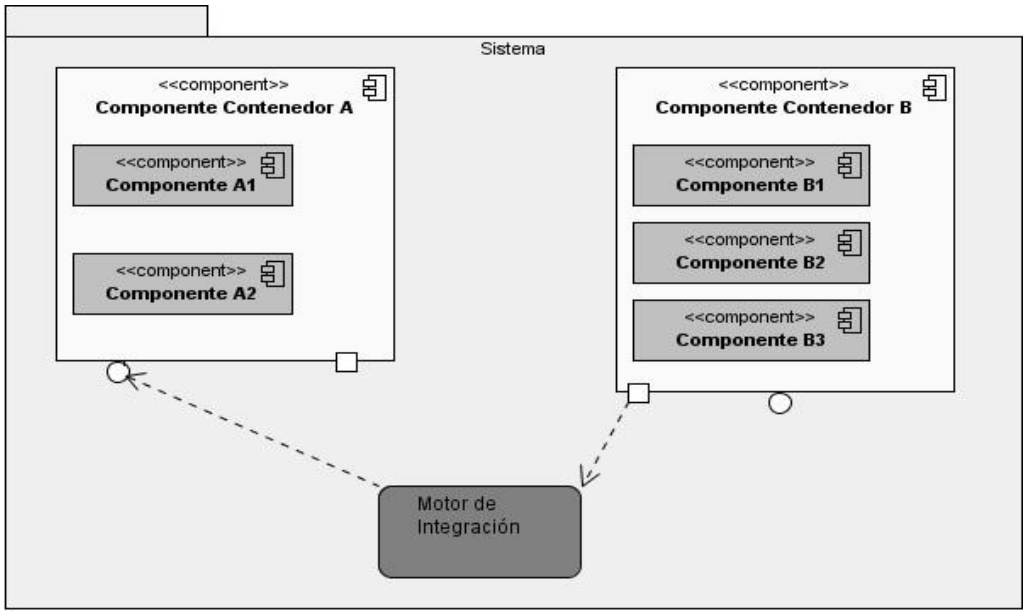


Figura 8. Interacción entre componentes contenedores.

La interacción entre componentes contenedores se realizará de forma muy similar a la integración interna o de primer nivel, haciendo uso del mismo motor de integración descrito en el primer nivel de interacción, ver figura 8.

Aquí vale aclarar que el componente contenedor contendrá aparte de los componentes nivel 1,

dos paquetes de entidades, referentes a “Servicios” y “Consumidor” y exactamente con el mismo objetivo y funcionamiento que en los componentes nivel 1.

De esta forma el movimiento de un componente nivel 1 a formar parte de un componente contenedor no lleva prácticamente ningún cambio de codificación en las entidades del componente nivel 1, solo en los paquete de entidades “Servicios” y “Consumidor” del componente contenedor, donde se deben implementar los contratos de los servicios que brindarán o consumirán los componentes internos presentes en el componente contenedor.

En la figura 8, se muestra la interacción entre dos componentes contenedores, usando exactamente la misma forma de interacción explicada en la interacción nivel 1. El componente contenedor A está compuesto por dos componentes nivel 1, más los dos paquetes ya explicados con anterioridad, brinda un servicio que es consumido por el componente contenedor B, con semejante estructura.

De esta forma una interacción entre componentes nivel 1, que se encuentre en componentes contenedores distintos, solo es posible realizarse por la interacción entre los componentes contenedores correspondientes.

2.4 Modelo de proceso

2.4.1 Organización estructural del equipo

Antes de presentar el flujo de actividades metodológicas para la definición de los componentes de sistema, se considera necesario presentar la estructura organizativa en la que está dispuesto el equipo encargado del desarrollo del sistema integral de gestión, ya que se considera significativa la relación existente entre la estructura organizativa y el desarrollo de este importante flujo de actividades.

La estructura consiste en un conjunto de líneas de desarrollo, con una estructura bien definida, las cuales contendrán un número de factorías encargadas de la implementación de uno o más proyectos. Tanto en las direcciones centrales como en las líneas de desarrollo existen roles y responsabilidades bien delimitadas y una estrecha relación en conjunto.

La responsabilidad del desarrollo de los proyectos radica en las líneas de producción y sus factorías correspondientes, el conjunto de direcciones centrales funge como rectores de temas metodológicos, así como de políticas de estandarización, aprobación y consolidación de información, ver anexo 3.

Fuera de las líneas de desarrollo se realizan actividades dentro del flujo de desarrollo del software

con carácter general y con equipos multidisciplinarios con el propósito de fomentar la reutilización de activos de software incluyendo el conocimiento. Estas actividades son las primeras actividades del análisis de las soluciones y la definición de los componentes del sistema, luego se continúa con el proceso de desarrollo definido en el interior de cada línea designada a implementar los componentes definidos.

El anexo 3, muestra la estructura explicada anteriormente. Cada factoría aunque no se grafica en el anexo, tiene una estructura interna de roles, pero esta varía según el objetivo de su creación, ya que sus funciones pueden variar entre:

- Desarrollo de componentes del sistema, definido con anterioridad por el equipo correspondiente.
- Mantenimiento ha determinado componente nivel 1, o componentes contenedor, que se encuentre en un determinado tipo de prueba.
- Desarrollo de determinados componentes nivel 1 de tipo integración, para una verticalización del sistema.

La función que se le asigne a la factoría determinará su composición. Por tales motivos las factorías son estructuras dinámicas en cuanto a su composición y que se destruyen luego de cumplido su objetivo o función.

Es válido aclarar que conceptualmente las definiciones de líneas de desarrollo y factorías de desarrollo, no se refieren a los modelos de "fábricas de software" tradicionales, referenciados en las bibliografías sobre el tema.

2.4.2 Guía metodológica para la definición de los componentes del sistema

Una vez explicada la estructura de organización, se da paso a presentar la guía de actividades metodológicas para la definición y formalización de los componentes de software. En la descripción de las actividades a continuación se hace referencia al conjunto de roles involucrados y a los artefactos que se generan o se hace uso de ellos, estos componentes esenciales de la guía serán explicados en el epígrafe 2.5. La guía metodológica contiene un total de once actividades y siete artefactos de entradas.

Actividad #1: Análisis de los artefactos generados en las fases de modelado de negocio y análisis del sistema.

Descripción: Estudio y análisis a profundidad de los artefactos generados en las fases iniciales.

Roles participantes: Arquitecto de sistema, Arquitecto de datos, Analista principal, Arquitecto de

componentes.

Artefactos de entrada: Mapa y descripción de los macro procesos, Modelo conceptual, Listado especificación y priorización de los requisitos, Restricciones del diseño (referidas a las fases iniciales), Glosario de términos, Visión del proyecto, Árbol de utilidades de los atributos de calidad.

Actividad #2: Estudio de reutilización de los componentes existentes en el repositorio.

Descripción: Permite identificar en qué medida las funcionalidades presentes a desarrollar ya están contenidas en los componentes desarrollados que se encuentran en el repositorio de componentes.

Roles participantes: Arquitecto de sistema, Arquitecto de datos, Arquitecto de componentes, Analista principal.

Artefactos de entrada: Listado especificación y priorización de los requisitos, Restricciones del diseño, Glosario de términos, Documentación asociada a los componentes presentes en el repositorio.

Artefactos de salida: Listado de componentes candidatos a reutilizar presentes en el repositorio de componentes.

Actividad #3: Estudio de factibilidad de los componentes candidatos a reutilización.

Descripción: Se realiza un estudio de la factibilidad de reutilización de los componentes definidos como candidatos, se define el escenario de reutilización:

- Incorporación del componente al paquete de la solución en desarrollo.
- Incorporación del componente al paquete de la solución en desarrollo con determinado ajuste en sus funcionalidades.
- Establecimiento de una integración con el componente ya que formará parte de la solución una vez listo su desarrollo para el escenario de despliegue candidato, por tanto no hay que incorporarlo a la vista de sistema del nuevo producto en la fase de desarrollo.

Roles participantes: Arquitecto de sistema, Arquitecto de datos, Arquitecto de componentes, Analista principal.

Artefactos de entrada: Listado especificación y priorización de los requisitos, Restricciones del diseño, Árbol de utilidad de los atributos de calidad, Listado de componentes candidatos a reutilizar.

Artefactos de salida: Listado de componentes a reutilizar en la nueva solución y bajo qué esquema de reutilización de los definidos.

Actividad #4: Identificar estructura de empaquetamiento.

Descripción: Permite identificar y delimitar responsabilidades de negocio a nivel macro con vistas a agrupar futuras funcionalidades en componentes contenedores y componentes nivel 1. Actividad orientada a favorecer el mantenimiento, la adaptabilidad y la flexibilidad.

Roles participantes: Arquitecto de sistema, Arquitecto de datos, Arquitecto de componentes, Analista principal.

Artefactos de entrada: Mapa de procesos, Listado y descripción de los macro procesos.

Artefactos de salida: Primera aproximación de los componentes contenedores.

Actividad #5: Análisis de los atributos de calidad y escenarios arquitectónicos.

Descripción: En esta actividad se persigue el análisis de los atributos de calidad (AC) a favorecer con el modelo de la solución. Además se identifican los escenarios arquitectónicos de mayor trascendencia (EA), todo a partir de los requisitos de software (RS) y restricciones del diseño (RD). Pueden aparecer nuevas RD como parte de decisiones arquitectónicas tomadas.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Analista principal, Arquitecto de datos.

Artefactos de entrada: Listado de requisitos, Restricciones del diseño, Árbol de utilidad de los atributos de calidad.

Artefactos de salida: Matriz de RD, EA, AC.

Actividad #6: Agrupamiento de los requisitos funcionales.

Descripción: La actividad tiene como responsabilidad realizar un agrupamiento semántico de los requisitos funcionales, identificándose la asignación de responsabilidades, los niveles de empaquetamiento, y las estructuras capaces de garantizar la mejor reutilización y colaboración posible.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Analista principal.

Artefactos de entrada: Listado de requisitos, Especificación de requisitos, Mapa de proceso del negocio.

Artefactos de salida: Aproximación de los componentes contenedores y componentes nivel 1 de la solución modelada.

Actividad #7: Formalización de los componentes del sistema.

Descripción: La actividad tiene como responsabilidad identificar y diseñar la configuración de los elementos de mayor abstracción de la Arquitectura de Sistema (Componentes contenedores y

Componentes nivel 1). Se realiza además la identificación de las entradas y salidas de cada elemento, las dependencias e integraciones, y los procesos de notificación a realizar en las principales acciones de colaboración para que sean posteriormente gestionadas y desarrolladas por el equipo de Arquitectura de Integración.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Analista principal, Arquitecto de datos.

Artefactos de entrada: Artefactos salida de la actividad 6, Listado de requisitos, Especificación de requisitos, Mapa de proceso del negocio.

Artefactos de salida: Documento especificación componentes contenedores nivel1, matriz integración de componentes nivel1, especificación de propiedades de componentes nivel1. Actualización de los artefactos: Especificación de los componentes contenedores, especificación de propiedades de componentes contenedores, matriz integración general, matriz de RD, EA, AC.

Actividad #8: Clasificación de los componentes según su función en el sistema.

Descripción: La actividad tiene como responsabilidad identificar y clasificar todos los componentes que conformarán la solución a desarrollar, se clasifican los componentes formalizados en la actividad anterior según su función en el sistema (categoría definida y explicada con anterioridad), se identifican otros tipos de componentes necesarios para el desarrollo de la solución en cuestión (componentes tecnológicos, de pruebas, de integración).

Roles participantes: Arquitecto de sistema, Arquitecto de datos, Arquitecto de componentes.

Artefactos de entrada: Artefactos de salida de la actividad 7.

Artefactos de salida: Actualización de Documento especificación de componentes nivel1.

Actividad #9: Priorización de los componentes y definición de las iteraciones de desarrollo.

Descripción: La actividad tiene como responsabilidad realizar el cálculo de la significación arquitectónica (cálculo de la complejidad, tamaño y criticidad) de los componentes del sistema, con ello su impacto en la arquitectura y la prioridad de implementación de los mismos, la información obtenida de esta actividad de la guía sirve de base para decidir qué elementos arquitectónicos incluir en cada posible iteración de desarrollo. Aquí hay que tener en cuenta la prioridad de los requisitos asociados al componente, la clasificación del componente y la criticidad del mismo.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Jefe de Línea.

Artefactos de entrada: Artefactos de salida de la actividad 7 actualizados en actividad 8.

Artefactos de salida: Actualización de Documento especificación de componentes nivel1.

Actividad #10: Discusión del plan de estimación y planificación del proyecto.

Descripción: La actividad tiene como responsabilidad realizar un análisis de la capacidad técnica del equipo, la disponibilidad tecnológica y productiva y se propone un primer boceto del cronograma de implementación y desarrollo de la línea base.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Jefe de Línea.

Artefactos de entrada: Todos los documentos generados hasta el momento, Visión del proyecto.

Artefactos de salida: Se actualiza el cronograma general del proyecto en desarrollo (artefacto independiente de la guía metodológica que se propone).

Actividad #11: Revisión Integral del diseño.

Descripción: La actividad tiene como responsabilidad el refinamiento de los artefactos generados en la fase de diseño. Es una revisión completa y profunda con el objetivo de llevar a la implementación el diseño lo mejor y más comprensible posible.

Roles participantes: Arquitecto de sistema, Arquitecto de componentes, Arquitecto de datos, Jefe de Línea.

Artefactos de entrada: Todos los documentos generados hasta el momento.

Artefactos de salida: Aprobación de todos los documentos generados hasta el momento, como parte de la obtención de los componentes del proyecto.

Al finalizar el desarrollo de un componente por parte de la línea de desarrollo y bajo el criterio del equipo de calidad del proyecto, el mismo se incorporará al repositorio de componentes con toda la documentación generada durante el proceso de diseño explicado.

La documentación asociada al componente formará parte del repositorio desde la culminación de la última actividad de la guía para la obtención de los mismos, y esa documentación se irá actualizando, si fuera necesario, en la medida que se implemente el componente.

2.5 Modelo del equipo

2.5.1 Roles en el proceso de obtención de los componentes

En la guía metodológica que se propone para la obtención de los componentes del sistema, están presentes un conjunto de roles con responsabilidades bien definidas, que tributan al correcto desarrollo y documentación de la arquitectura de sistema a desarrollar. A continuación se describen.

Arquitecto de Sistema

Responsabilidades

- Dirige todas las actividades del proceso de obtención de los componentes del sistema.
- Define los escenarios arquitectónicos de mayor trascendencia presentes en el desarrollo del sistema.
- Aprueba y evalúa las decisiones arquitectónicas de sistema en las líneas de producción.
- Aprueba los cambios en la línea base del sistema desde el punto de vista arquitectónico.
- Participa en las decisiones de interoperabilidad e intraoperabilidad del sistema.
- Participa en las definiciones de las restricciones de diseño y los atributos de calidad a favorecer en el producto desde la solución arquitectónica de sistema.
- Elabora los documentos rectores y de control necesarios para el aseguramiento de los lineamientos de calidad desde el punto de vista de la arquitectura de sistema.

Arquitecto de Componentes

Responsabilidades

- Máximo responsable por el cumplimiento de las disposiciones arquitectónicas definidas para los componentes a desarrollar en la línea.
- Máximo responsable por el diseño interno de los componentes definidos a desarrollar en la línea.
- Elabora los documentos rectores y de control necesarios para el aseguramiento de los lineamientos de calidad desde el punto de vista de la arquitectura de los proyectos de la línea.
- Máximo responsable por la gestión de los documentos descriptivos de los componentes desarrollados en la línea y su incorporación al repositorio de componentes.
- Vela por el cumplimiento de la línea base de los proyectos desde las soluciones arquitectónicas.

Arquitecto de Datos

Responsabilidades

- Define los estándares de codificación y diseño de bases de datos.
- Define el diseño desde el punto de vista de la persistencia de los datos, de los componentes que lo requieran.
- Define y gestiona el método de actualización y control de versiones de la base de datos.
- Define la metodología de trabajo de la arquitectura de datos y la estrategia de soporte.

- Construye y documenta la vista de datos.
- Define las pautas para el reflejo necesario en la vista de datos de las integraciones del sistema.

Analista Principal

Responsabilidades

- Participa con el cliente y el usuario final recogiendo las entradas de los involucrados relevantes.
- Realiza el modelado de los procesos a informatizar.
- Realiza el modelo conceptual del análisis del sistema a desarrollar.
- Realiza la especificación de requisitos.
- Realiza la priorización de los requisitos identificados según la visión del proyecto y las características del sistema a desarrollar.
- Identifica y documenta las restricciones del diseño, desde la dimensión del modelado del negocio y análisis, presentes en el sistema a desarrollar.
- Construye el glosario de los términos y conceptos propios de las áreas funcionales a incidir con el sistema.
- Documenta los flujos de integración existes en el sistema a desarrollar.
- Realiza el seguimiento de los requisitos durante todo el desarrollo del proyecto.
- Participa en el diseño de la solución.
- Participa en la identificación y diseño de los componentes de tipos pruebas a desarrollar en el sistema.
- Participa en la elaboración del Plan de Administración de Requisitos.

Jefes de Líneas de Desarrollo

Responsabilidades

- Garantizar los cronogramas y compromisos del desarrollo de los componentes definidos a construirse en la línea de desarrollo.
- Supervisar el proceso de desarrollo.
- Máximo responsable de la implementación correcta al tratamiento de cada una de las áreas de conocimiento de la gestión de proyecto, dentro de los proyectos en desarrollo en su línea.
- Organizar y controlar el trabajo de los miembros de su línea.
- Controlar los indicadores de eficiencia.

- Velar por el cumplimiento del modelo y metodología de desarrollo de software seleccionado a utilizarse en el proyecto.
- Máximo responsable por el cumplimiento de los indicadores de calidad a medírsele a cada componente desarrollado en la línea correspondiente.

2.5.2 Artefactos

Como parte del proceso de obtención de los componentes del sistema que se desarrolla, se propone un conjunto de artefactos a generarse durante el desarrollo de las actividades de la guía metodológica. Basado en la estructura organizacional del equipo de desarrollo, explicada anteriormente, los artefactos propuestos se dividen en dos grandes grupos:

- Los artefactos que forman parte de la línea base de la arquitectura del sistema integral de gestión (ERP), referentes a la documentación de los componentes contenedores del sistema.
- Los artefactos que forman parte de la línea base de la arquitectura de los sistemas desarrollados en cada una de las líneas, referentes a la documentación de los componentes nivel 1.

Los artefactos pertenecientes al primer grupo quedan explicados a continuación:

Artefacto # 1: MATRIZ ANÁLISIS DE ATRIBUTOS DE CALIDAD - RESTRICCIONES DEL DISEÑO.

Descripción: Contiene una vista y descripción global de las restricciones del diseño identificadas que aseguran el fortalecimiento de determinado atributo de calidad por cada componente contenedor identificado. Por cada restricción se especifica:

- Código.
- Complejidad de implementación.
- Atributo de Calidad que fortalece.
- Componente contenedor que la implementará.

La complejidad será un número entre 1 y 10, siendo 10 la mayor complejidad posible, y el número será designado basado en análisis realizado entre los equipos de arquitectos y analistas en conjunto. (Ver anexo 4).

Momento de generación del artefacto: Se tiene con anterioridad antes de comenzar las actividades descritas en la presente guía, la relación restricciones de diseño – atributos de calidad, referidas a las restricciones identificadas en las fases de modelado de negocio y análisis, y se actualiza dicha relación incorporando las restricciones de diseño identificadas en el diseño arquitectónico y se incorpora a la relación los componentes contenedores identificados en la actividad 4 de la guía.

Roles responsables: Arquitecto de sistema y Analista principal.

Artefacto # 2: ESPECIFICACIÓN DE COMPONENTES CONTENEDORES.

Descripción: Contiene un análisis detallado de cada uno de los componentes contenedores, la descripción de cada uno de ellos, el código de identificación que tendrá el componente contenedor dentro de la línea base. El conjunto de servicios que brinda cada componente de este tipo, y de cada uno de los servicios el contrato que acompaña a este (Ver anexo 5), desglosado en:

- Descripción del servicio.
- Parámetros de entrada del servicio.
- Parámetros de salida del servicio.

Momento de generación del artefacto: Se inicia la generación del artefacto a partir de la actividad 4 de la guía y luego se va actualizando posteriormente.

Roles responsables: Arquitecto de sistema.

Artefacto # 3: ESPECIFICACIÓN DE LAS PROPIEDADES DEL COMPONENTE CONTENEDOR.

Descripción: Contiene un conjunto de información de todos los componentes contenedor, con el objetivo de lograr obtener las propiedades de tamaño, complejidad y criticidad del componente, (Ver anexo 6), queda constituido de la siguiente manera, por cada componente se medirá:

- Cantidad de requisitos funcionales.
- Complejidad promedio de los requisitos funcionales (valor entre 1 y 10).
- Cantidad de restricciones del diseño.
- Complejidad promedio de las restricciones (valor entre 1 y 10).
- Cantidad de servicios que brinda el componente.
- Cantidad de componentes dependientes del mismo.

Con todos estos datos se pueden calcular las tres propiedades a medir mencionadas con anterioridad:

Tamaño del componente

$$TC = \frac{Kr * 2 + Krd * 2 + Ks}{100}$$

Donde:

TC -> Tamaño del componente.

Kr -> Cantidad de requisitos funcionales.

Krd-> Cantidad de restricciones del diseño.

Ks-> Cantidad de servicios que brinda el componente.

Complejidad del componente

$$CoC = \frac{Cr * 2 + Crd * 2 + TC}{100}$$

Donde:

Coc -> Complejidad del Componente.

Cr -> Complejidad promedio de los requisitos funcionales.

Crd -> Complejidad promedio de las restricciones del diseño.

TC-> Tamaño del componente.

Criticidad del Componente

$$CrC = \frac{Kcd * 2 + \sum KcdD}{100}$$

Donde:

Crc -> Criticidad del Componente.

Kcd -> Cantidad de componentes dependientes directos.

KcdD-> Cantidad de componentes dependientes de los dependientes directos del componente.

Momento de generación del artefacto: Se inicia la generación del artefacto a partir de la actividad 3 de la guía y luego se va actualizando posteriormente.

Roles responsables: Arquitecto de sistema y arquitecto de componente.

Artefacto # 4: Documento Matriz de Integración del Sistema Integral de Gestión.

Descripción: Contiene una matriz bidimensional de integración (dependencias), entre los componentes contenedor del sistema integral, especificándose los puntos de contactos entre cada uno de estos componentes y mediante qué servicios ocurre la interacción (Ver anexo 7).

Momento de generación del artefacto: Se inicia la generación del artefacto a partir de la actividad 3 de la guía y luego se va actualizando posteriormente.

Roles responsables: Arquitecto de sistema.

Los artefactos pertenecientes al segundo grupo, se exponen a continuación:

Artefacto # 1: DOCUMENTO DE ESPECIFICACIÓN DE LOS COMPONENTES.

Descripción: Contiene un análisis detallado de cada uno de los componentes, la descripción de cada uno, el código de identificación que tendrá el componente dentro de la línea base. El conjunto

de servicio que brinda cada componente, y de cada uno de los servicios el contrato que acompaña a este, (Ver anexo 8), desglosado en:

- Descripción del servicio.
- Parámetro de entrada del servicio.
- Parámetros de salida del servicio.

Momento de generación del artefacto: Se inicia la generación del artefacto a partir de la actividad 4 de la guía y luego se va actualizando posteriormente.

Roles responsables: Arquitecto de sistema, Arquitecto de componentes.

Artefacto # 2: ESPECIFICACIÓN DE LAS PROPIEDADES DEL COMPONENTE.

Descripción: Contiene un conjunto de información de todos los componentes, con el objetivo de lograr obtener las propiedades de tamaño, complejidad y criticidad del componente, queda constituido de la siguiente manera, por cada componente se medirá un conjunto de indicadores (Cantidad de requisitos funcionales, Complejidad promedio de los requisitos, Cantidad de restricciones del diseño, Complejidad promedio de las restricciones, Cantidad de servicios que brinda el componente, Cantidad de componentes dependientes del mismo)(Ver anexo 9).

Con todos estos datos se pueden calcular las tres propiedades a medir mencionadas con anterioridad:

Tamaño del componente

$$TC = \frac{Kr * 2 + Krd * 2 + Ks}{100}$$

Complejidad del componente

$$CoC = \frac{Cr * 2 + Crd * 2 + TC}{100}$$

Criticidad del Componente

$$CrC = \frac{Kcd * 2 + \sum KcdD}{100}$$

Siendo estas las mismas fórmulas empleadas para calcular las propiedades en los componentes contenedores, explicadas en el artefacto "ESPECIFICACIÓN DE LAS PROPIEDADES DEL COMPONENTE CONTENEDOR", teniendo las variables empleadas el mismo significado conceptual.

Momento de generación del artefacto: Se inicia la generación del artefacto a partir de la actividad 4

de la guía y luego se va actualizando posteriormente.

Roles responsables: Arquitecto de sistema y arquitecto de componente.

Artefacto # 3: MATRIZ DE INTEGRACIÓN DE LOS COMPONENTES NIVEL 1.

Descripción: Contiene una matriz bidimensional de integración (dependencias), entre los componentes nivel 1 pertenecientes al componente contenedor, especificándose los puntos de contactos entre cada uno de estos componentes y mediante qué servicios ocurre la interacción (Ver anexo 10).

Roles responsables: Arquitecto de sistema y arquitecto de componentes.

Queda bajo la responsabilidad del arquitecto de sistema y de los arquitectos de componentes de las líneas, las constantes actualizaciones y cumplimiento de cada una de las definiciones que se establecen al llenar los artefactos que se han presentado, tanto en el desarrollo del flujo de las actividades metodológicas en la obtención de los componentes del sistema de cada uno de los proyectos, como en el posterior flujo de actividades previstas según la metodología de desarrollo utilizada en el proceso de desarrollo de software de los proyectos en las líneas.

2.6 Conclusiones Parciales

A partir de los elementos teóricos estudiados se elaboró una propuesta metodológica para la obtención de los componentes de sistema en los proyectos de software del programa ERP-Cuba,

Concluyendo que:

- La guía propuesta permite la obtención de los componentes del sistema basado en la reutilización de componentes desarrollados con anterioridad, impactando directamente en los tiempos de desarrollo de los distintos proyectos que conforman el programa ERP-Cuba.
- La estructura de empaquetamiento definida para el sistema, así como la forma de interacción entre los distintos componentes, incide directamente en una serie de indicadores y atributos que son reflejados en la calidad final del sistema.
- La estructura organizativa que requiere la aplicación de la guía metodológica propuesta se ajusta sin mayores dificultades a la estructura existente en el equipo de desarrollo del programa ERP-Cuba.

CAPÍTULO 3. ANÁLSIS DE RESULTADO

3.1 Introducción

En el capítulo se realiza el análisis de los resultados de la puesta en práctica de la presente guía en 13 proyectos de la población declarada. Los resultados obtenidos se brindarán en función de un conjunto de indicadores demostrativos de la disminución de los tiempos de desarrollo de los proyectos. Se mostraran los resultados de la ejecución del experimento puesto en práctica, haciendo uso de métodos estadísticos para el procesamiento de los resultados obtenidos.

Finalmente se hace una valoración del impacto de la guía en la calidad, de los proyectos aplicados.

3.2 Indicadores de medición

Para el análisis de los resultados de la puesta en práctica de la presente investigación, se toman como variable demostrativa la disminución de los tiempos de desarrollo de los proyectos seleccionados.

Esta variable se medirá en función de una serie de indicadores definidos.

Disminución de los tiempos de desarrollo: Esta variable se medirá en función de la reutilización de componentes desarrollados.

Indicadores:

- Cantidad de componentes reutilizados.
- Cantidad de instancias de reutilización.

3.2.1 Características de la muestra

La población total está constituida por todos los proyectos del programa ERP-Cuba, para un total de diecisiete. 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 que por su estado dentro del proceso de desarrollo, permitiera una mejor evaluación de la variable de la presente investigación. La muestra quedó constituida por trece proyectos, lo que representa un setenta y seis por ciento de la población total.

Todos estos proyectos presentan características similares, todos forman parte del programa ERP-Cuba, presentan negocios similares y con altos niveles de integración entre ellos, la mayoría ha tenido que recibir verticalización para conformar soluciones específicas de cara al cliente, y todos comparten la misma arquitectura de sistema base, conceptualmente.

Los proyectos seleccionados y consta en acta de inscripción en la dirección de la universidad, son:

Tabla 2. Proyectos del programa ERP-Cuba, pertenecientes a la muestra seleccionada.

NOMBRE DEL PROYECTO	SIGLAS	FASE EN LA QUE SE ENCUENTRA
Planificación por objetivos	**PPO**	**Despliegue**
Seguridad	**Acaxia**	**Despliegue**
Maco de Trabajo	**Sauxe**	**Despliegue**
Configuración del sistema	**Conf.**	**Pruebas de aceptación**
Contabilidad	**Cont.**	**Pruebas de aceptación**
Costos y Procesos	**Cost.**	**Pruebas de aceptación**
Caja	**Caj.**	**Pruebas de aceptación**
Banco	**Banc.**	**Pruebas de aceptación**
Cobros y Pagos	**COPA**	**Pruebas de aceptación**
Capital Humano	**RRHH**	**Pruebas de aceptación**
Estructura y Composición	**ESTRUCT.**	**Pruebas de aceptación**
Auditoría	**AUD.**	**Pruebas de liberación interna**
Inventario	**INV.**	**Pruebas de aceptación**

Los proyectos que se encuentran en fase de despliegue y pruebas de aceptación, significa que ya han sido certificados por CALISOFT, 12 en este caso, por tanto han transitado por el ciclo definido para los productos del centro CEIGE, luego de culminar la fase de implementación.

- Fase de implementación.
- Pruebas de liberación interna.
- Pruebas piloto.
- Pruebas de liberación.
- Pruebas de aceptación
- Despliegue.

3.3 Evaluación

3.3.1 Análisis de reutilización

Por las características de los proyectos de la muestra y el tiempo de explotación de la presente propuesta metodológica para la obtención de los componentes del sistema, se pueden identificar tres formas de reutilización en los proyectos del programa ERP-Cuba.

Reutilización Total (RT): Ocurre cuando un componente nivel 1, es reutilizado en el desarrollo de un componente contenedor, de forma total y sin ningún tipo de ajuste o personalización, es incorporado el componente reutilizable en la solución, sin gasto de tiempo para su explotación.

Reutilización Total con ajustes de servicios (RTA): Ocurre cuando un componente nivel 1, es reutilizado en el desarrollo de un componente contenedor, de forma total y al hacer las integraciones pertinentes para la incorporación a la solución, son redefinidas o ajustadas algunas funcionalidades.

Reutilización Parcial (RP): Ocurre cuando un componente nivel 1, es reutilizado en el desarrollo de un componente contenedor, de forma parcial, solo algunas de sus funcionalidades y es modificado y ajustado el componente reutilizable por extensión (se le incorporan nuevas funcionalidades) ó por disminución (se le eliminan una serie de funcionalidades y se estructura nuevamente el componente con el alcance deseado).

La propuesta metodológica propone dos dimensiones de clasificación de los componentes (Ver 2.3.2), bajo esta taxonomía se establece que la reutilización llevada a cabo en los proyectos del programa ERP-Cuba, es "Componentes contenedores reutilizan para su desarrollo o empaquetamiento los componentes nivel 1", de ahí se concluye que los componente propensos a la reutilización es el universo de los componentes nivel 1 desarrollados y los componentes propensos a reutilizar es el universo de los componentes contenedores desarrollados.

En la muestra seleccionada se encuentran un total de 11 componentes contenedores y el número de componentes nivel 1 por cada uno de ellos se muestra en la tabla 3.

Tabla 3. Relación componentes contenedores – componentes nivel 1.

Componente Contenedor	**Cantidad de componentes nivel 1**
CC_ PPO	3
CC_ Marco_Trabajo	9
CC_ Configuración	21
CC_ Contabilidad	7
CC_ Costos_Procesos	4
CC_ Caja	12
CC_ Banco	6
CC_ Cobros_Pagos	13
CC_ RRHH	15
CC_ Auditoría	7
CC_Inventarios	22

A la cantidad de componentes nivel 1, presente en la muestra, hay que sumarle los componentes CN_Seguridad y CN_Estructura, que son dos proyectos que su desarrollo desde el punto de vista arquitectónico se reduce al desarrollo de un solo componente nivel 1 de tipo núcleo respectivamente, para un total de 121 componentes nivel 1.

A continuación se muestra la relación de los componentes contenedores que han reutilizado componentes nivel 1 del tipo de reutilización RT.

Tabla 4. Instancias de reutilización tipo RT.

	CN_Glob	CN_IOC	CN_Nom	CN_Per	CN_Mtm.	CN_Eje
CC_ PPO	X	X	X			
CC_ Configuración		X	X		X	
CC_ Contabilidad	X	X		X	X	X
CC_ Costos_Procesos	X	X	X	X		X
CC_ Caja	X	X		X		X
CC_ Banco	X	X		X		X
CC_ Cobros_Pagos	X	X		X		X
CC_ RRHH	X	X	X			X
CC_ Auditoría	X	X				
CC_Inventarios	X	X		X		X

En la tabla 5 se muestra la relación de los componentes contenedores que han reutilizado componentes nivel 1 del tipo de reutilización RTA.

Tabla 5. Instancias de reutilización tipo RTA.

	CN_Cierre	CN_Form	CN_CliPro	CN_PafF	CN_UMed	CN_Conc	CN_Doc	CN_RrCont
CC_ PPO		X						
CC_ Configuración								
CC_ Contabilidad	X	X				X	X	X
CC_ Costos_Procesos	X	X						
CC_ Caja	X	X	X			X	X	X

CC_ Banco	X	X	X			X	X	X
CC_ Cobros_ Pagos	X	X	X	X		X	X	X
CC_ RRHH	X	X						
CC_ Auditoría		X						
CC_Inventarios	X	X	X		X	X	X	X

En la tabla 6 se muestra la relación de los componentes contenedores que han reutilizado componentes nivel 1 del tipo de reutilización RP.

Tabla 6. Instancias de reutilización de tipo RP.

	CN_CarIni	**CN_CompTip**	**CN_WorF**	**CN_Exp**
CC_ PPO	X		X	X
CC_ Configuración				X
CC_ Contabilidad	X	X		
CC_ Costos_Procesos	X			
CC_ Caja	X			
CC_ Banco	X			
CC_ Cobros_Pagos	X			
CC_ RRHH	X			
CC_ Auditoría				
CC_Inventarios	X			

Consolidando la información expuesta en las tres tablas anteriores se tiene que, existen un total de 87 instancias de reutilización, desglosado por los tipos de reutilización de la siguiente manera:

- RT – 38
- RTA - 37
- RP – 12

Se define la unidad de medida **"ciclo de trabajo (CT)**" como el tiempo de desarrollo empleado para implementar un componente nivel 1.

Basado en análisis de los cronogramas y en estadísticas de planificación de los proyectos de 12

meses de desarrollo, se pueden concluir los siguientes planteamientos bases:

Las instancias de reutilización tipo RT coinciden con la cantidad de CT ahorrado en desarrollo en los proyectos, por constituir una reutilización de componente sin gasto de tiempo. Las instancias de reutilización de tipo RTA, constituyen una reutilización con un gasto de tiempo pequeño, por tanto se asume basado en los datos históricos del proyecto, que el tiempo medio ahorrado por este tipo de reutilización significará 0,75CT. Las instancias de reutilización de tipo RP, constituyen una reutilización con un costo de tiempo significativo, pero siempre inferior a desarrollar el componente nuevamente, por tanto se asume que el tiempo medio ahorrado por este tipo de reutilización significará 0,50CT.

1 instancia de reutilización RT -> ahorra 1CT.

1 instancia de reutilización RTA -> ahorra 0,75CT.

1 instancia de reutilización RP -> ahorra 0,50CT.

Basado en el análisis anterior se plantea que el tiempo ahorrado por concepto de reutilización de componentes se expresará de la siguiente manera:

$$CTa = CTart + CTarta + CTarp$$

Donde:

CTa: Ciclos de trabajo ahorrado.

CTart: Ciclos de trabajo ahorrado por concepto de reutilización de tipo RT.

CTarta: Ciclos de trabajo ahorrado por concepto de reutilización de tipo RTA.

CTarp: Ciclos de trabajo ahorrado por concepto de reutilización de tipo RP.

Aplicando las conversiones de cada tipo de instancia de reutilización a CT y sustituyendo las CT obtenidas para cada tipo de instancia en la fórmula anterior, se procede a calcular el valor de la variable CTa.

CTa = 38 * 1 + 37 * 0,75 + 12 * 0,50

CTa = 71,75

Significa que en la muestra de proyectos seleccionados, por concepto de reutilización de componentes, se ha disminuido en 71,75 CT el tiempo de desarrollo de los proyectos.

3.3.2 Análisis de disminución del tiempo de desarrollo

Tomando como concepto la definición de la unidad de medida CT, como el tiempo de desarrollo

empleado para implementar un componente nivel 1 y teniendo la cantidad de componentes nivel 1 desarrollados en los proyectos de la muestra se presentan los siguientes datos:

En los trece proyectos de la muestra, se han desarrollado un total de 121 componentes nivel1, lo que significa que el sistema integral Cedrux, para la muestra tomada, ha demorado una cantidad de 121 CT por concepto de implementación de los componentes que conforman sus proyectos.

Como conclusión de 3.3.1, se conoce que por concepto de reutilización, para la muestra seleccionada, se ha ahorrado una cantidad de 71,75 CT, lo que significa que por concepto de implementación de los componentes que conforman los proyectos, el tiempo de desarrollo debía ser:

TDe = Trd + Tra

dónde:

TDe: Tiempo de desarrollo estimado.

Trd: Tiempo real de desarrollo.

Tra: Tiempo real ahorrado.

TDe = 121 CT + 71,75 CT

TDe = 192,75 CT

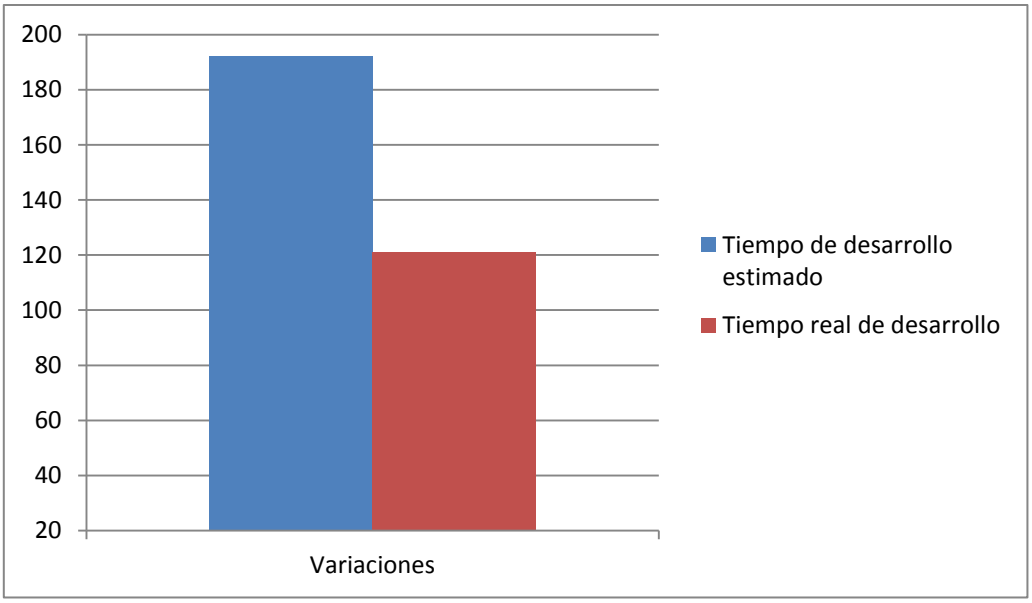


Figura 9. Variaciones entre el tiempo estimado y el tiempo real de desarrollo.

La gráfica anterior evidencia las diferencias entre el tiempo de desarrollo estimado y el empleado por concepto de reutilización.

Las fechas de desarrollo de los componentes que conforman los proyectos seleccionados, van desde septiembre 2008 hasta octubre 2010, lo que significa aproximadamente un total de 26 meses de desarrollo.

Asumiendo para simplificar el análisis los meses de 30 días, significa un estimado en días de 780 aproximadamente.

Convirtiendo la unidad de tiempo CT a días:

780 días significan 121 CT, que es el tiempo real de desarrollo expresado en CT.

Por tanto:

1CT = 6,44 días aproximadamente.

Por lo que se puede arribar a la conclusión que si 71,75 CT fue el tiempo ahorrado por concepto de reutilización, eso significa que por dicho concepto, fomentado por la propuesta metodológica presente, fue disminuido en 462 días aproximadamente el tiempo de desarrollo.

3.3.3 Ejecución del experimento

El experimento quedó diseñado de la siguiente manera:

Se utilizará como grupo experimental los proyectos:

Proyectos	**Nomenclatura**
Contabilidad	0
Costos y Procesos	0
Caja	0
Banco	0
Cobros y Pagos	0
RRHH	0
Auditoria	0
Inventarios	0
Planificación	0

Como grupo de control se utilizarán los proyectos:

Proyectos	**Nomenclatura**
Configuración	1
Estructura y Composición	1
Sauxe	1
Acaxia	1

Para las distintas comparaciones a realizar en la ejecución del experimento que validen los resultados del mismo, se presenta la siguiente figura:

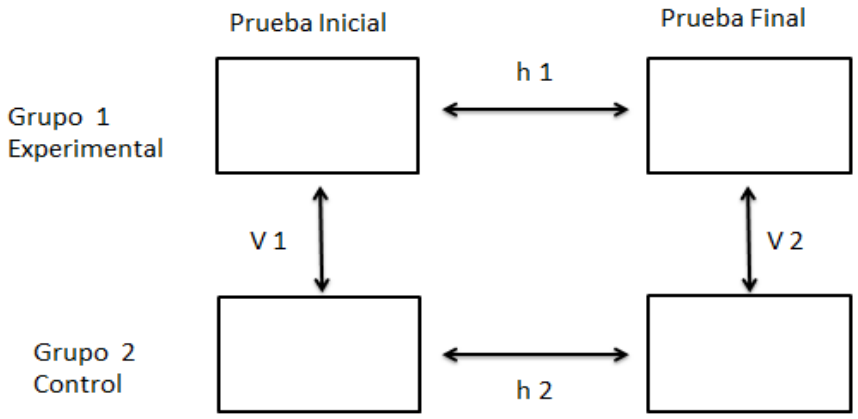


Figura 10. Tipos de pruebas para el análisis estadístico del experimento.

Para las distintas comparaciones se hará uso de los indicadores: tiempo estimado, tiempo real con reutilización y tiempo ahorrado.

Los valores de cada uno de los proyectos presentes en el experimento en función de los indicadores antes mencionados se presentan:

Tabla 7. Valores de los indicadores definidos por cada proyecto de la muestra.

Proyecto	**Grupo**	**Tiempo estimado(días)**	**Tiempo real con reutilización (días)**	**Tiempo ahorrado (días)**
Contabilidad	0	748,5	690	58.5
Costos y Procesos	0	732	690	42
Caja	0	834	780	54
Banco	0	834	780	54
Cobros y Pagos	0	834	780	54
RRHH	0	696	660	36
Auditoria	0	376,5	360	16.5
Inventarios	0	838,5	780	58.5
Planificación	0	373,5	360	13.5
Configuración	1	420	430	-10
Estructura y Composición	1	420	425	-5
Sauxe	1	360	367	-7
Acaxia	1	420	430	-10

Prueba estadística 1:

Aplicando el test de *Man-Whitney* para dos muestras independientes a la variable *Tiempo estimado*, se muestra los resultados en la siguiente figura

Mann-Whitney Test

Ranks

	Control	N	Mean Rank	Sum of Ranks
TiempoEstimado	0	9	8.33	75.00
	1	4	4.00	16.00
	Total	13		

Test Statistics(c)

			TiempoEstimado
Mann-Whitney U			6.000
Wilcoxon W			16.000
Z			-1.872
Asymp. Sig. (2-tailed)			.061
Exact Sig. [2*(1-tailed Sig.)]			.076(a)
Monte Carlo Sig. (2-tailed)	Sig.		.072(b)
	99% Confidence Interval	Lower Bound	.065
		Upper Bound	.079
Monte Carlo Sig. (1-tailed)	Sig.		.037(b)
	99% Confidence Interval	Lower Bound	.032
		Upper Bound	.042

Figura 11. Prueba estadística 1, test de *Man-Whitney* para dos muestras independientes.

Como se puede apreciar en el resultado señalado, no existe diferencia significativa entre los dos grupos en cuanto a la variable tiempo inicial, por lo que se puede concluir que al iniciar el experimento ambos grupos se encontraban en igualdad de condiciones.

Prueba estadística 2:

Aplicando el test de *Man-Whitney* para dos muestras independientes a la variable *Tiempo ahorrado*, luego de ejecutar el experimento, se muestra los resultados en la siguiente figura

Mann-Whitney Test

Ranks

	Control	N	Mean Rank	Sum of Ranks
TiempoAhorrado	0	9	9.00	81.00
	1	4	2.50	10.00
	Total	13		

Test Statistics(c)

			TiempoAhorrado
Mann-Whitney U			.000
Wilcoxon W			10.000
Z			-2.805
Asymp. Sig. (2-tailed)			.005
Exact Sig. [2*(1-tailed Sig.)]			.003(a)
Monte Carlo Sig. (2-tailed)	Sig.		.001(b)
	99% Confidence Interval	Lower Bound	.000
		Upper Bound	.002
Monte Carlo Sig. (1-tailed)	Sig.		.001(b)
	99% Confidence Interval	Lower Bound	.000
		Upper Bound	.002

Figura 12. Prueba estadística 2, test de *Man-Whitney* para dos muestras independientes.

Como se puede apreciar en los resultados señalados existe diferencia significativa entre los dos grupos en cuanto a la variable tiempo ahorrado. Se evidencia que la inclusión de la aplicación de la propuesta introdujo cambios significativos en los proyectos del grupo experimental determinados por un mayor ahorro del tiempo de desarrollo que en los proyectos del grupo de control.

Prueba estadística 3:

Aplicando el método estadístico de *Wilcoxon* para muestras apareadas, a las variables *Tiempo real con reutilización* y *Tiempo estimado* en el grupo experimental, se muestra los resultados en la siguiente figura

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
TiempoRealConReutilizacion - TiempoEstimado	Negative Ranks	9(a)	5.00	45.00
	Positive Ranks	0(b)	.00	.00
	Ties	0(c)		
	Total	9		

a TiempoRealConReutilizacion < TiempoEstimado

b TiempoRealConReutilizacion > TiempoEstimado

c TiempoRealConReutilizacion = TiempoEstimado

Test Statistics(b,c)

			TiempoRealConReutilizacion - TiempoEstimado
Z			-2.677(a)
Asymp. Sig. (2-tailed)			.007
Monte Carlo Sig. (2-tailed)	Sig.		.003
	99% Confidence Interval	Lower Bound	.002
		Upper Bound	.005
Monte Carlo Sig. (1-tailed)	Sig.		.002
	99% Confidence Interval	Lower Bound	.001
		Upper Bound	.003

Figura 13. Prueba estadística 3, test de *Wilcoxon* para dos muestras apareadas.

Como se puede apreciar en los resultados señalados existe diferencia significativa entre las dos variables evaluadas, el valor obtenido está muy cercano al deseado, esto significa que hubo avance, en el grupo experimental, desde el inicio de la prueba hasta el final.

Prueba estadística 4:

Aplicando el método estadístico de *Wilcoxon* para muestras apareadas, a las variables *Tiempo real con reutilización* y *Tiempo estimado* en el grupo de control, se muestra los resultados en la siguiente figura

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
TiempoRealConReutilizacion - TiempoEstimado	Negative Ranks	1(a)	3.50	3.50
	Positive Ranks	3(b)	2.17	6.50
	Ties	0(c)		
	Total	4		

a TiempoRealConReutilizacion < TiempoEstimado

b TiempoRealConReutilizacion > TiempoEstimado

c TiempoRealConReutilizacion = TiempoEstimado

Test Statistics(b,c)

			TiempoRealConReutilizacion - TiempoEstimado
Z			-.557(a)
Asymp. Sig. (2-tailed)			.577
Monte Carlo Sig. (2-tailed)	Sig.		.750
	99% Confidence Interval	Lower Bound	.739
		Upper Bound	.761
Monte Carlo Sig. (1-tailed)	Sig.		.377
	99% Confidence Interval	Lower Bound	.365
		Upper Bound	.390

Figura 14. Prueba estadística 4, test de *Wilcoxon* para dos muestras apareadas.

Como se puede apreciar en los resultados señalados no existen diferencias significativas entre las dos variables evaluadas, esto significa que no hubo variaciones en cuanto a las variables en cuestión, en el grupo de control, desde el inicio de la prueba hasta el final.

3.4 Impacto

El desarrollo del sistema Cedrux surgió como una necesidad de la dirección del país en aras de:

- Sustituir los pagos de licencia de este tipo de software, por concepto de importaciones que hace hoy el país en varios sectores empresariales.
- Contar con un sistema capaz de responder a las peculiaridades del sistema financiero contable cubano.
- Contar con un único sistema de gestión integral en las entidades cubanas, que permitiese la necesaria comunicación entre las entidades.

La aplicación de la presente propuesta metodológica incidió de manera positiva en los tiempos de desarrollo del sistema (demostrado en epígrafes anteriores).

El programa ERP-Cuba inició sus proyectos de desarrollo de software en septiembre de 2008 y fue posible en un período de 5 meses tener una primera versión funcional de 6 de los subsistemas que conforman hoy el sistema integral, un tiempo bastante pequeño para el desarrollo de seis soluciones integradas en este tipo de software. Un mes más tarde, en febrero de 2009 fue posible iniciar las primeras pruebas piloto en 6 entidades nacionales (45):

- ICID
- Yuri Gagarin
- Centro de Gestión
- Hospital Naval
- Rafael Trejo
- UCI

Luego a medida que el sistema fue madurando en estabilidad, se ampliaron las entidades en pruebas piloto a:

- MINFAR (área logística)
- MINFAR (área contable)
- CubaTaxi

Así como el número de soluciones, a 11 subsistemas integrados.

Adicionalmente, la solución se entregó en algunas de las empresas de software del país como parte de un paquete de Cedrux con el objetivo de verticalizar el producto para nuevos destinos de implantación en el ámbito nacional. Entre ellos se pueden mencionar: Desoft y el Centro de desarrollo de Holguín (45).

El proceso de reutilización de componentes desarrollado dentro del sistema ha incidido de manera significativa en que se pueda contar hoy con 17 soluciones integradas terminadas o en fase de terminación en un período de 26 meses, manteniendo por aproximadamente 1 año un número de soluciones en pruebas pilotos en un conjunto de entidades que han oscilado entre 6 y 8 y un despliegue piloto de algunas de las soluciones en el MINFAR.

La ventaja principal de la solución radica en la posibilidad de adaptación del sistema en varios entornos de negocio en el conjunto de empresas cubanas, garantizando que la misma solución cubra la gran mayoría de los contextos de despliegue. A pesar de ser concebida inicialmente para el

entorno nacional, ya se han identificado potencialidades que le permiten ampliar su destino de implantación (45).

3.4.1 Análisis costo beneficio

Actualmente el centro CEIGE trabaja en concretar convenios que implican un incremento significativo en los ingresos por conceptos de exportación.

Desde el punto de vista del análisis costo-beneficio se deben citar los convenios para proyectos de exportación en los que el CEIGE trabaja actualmente mediante la comercialización de Cedrux, convenios que implican un incremento significativo en los ingresos por conceptos de exportación. Actualmente, con Venezuela, se encuentran contratados y en desarrollo un sistema de mantenimiento y un sistema de gestión de medicamentos, sobre la tecnología de Cedrux (45).

Ambos proyectos calculan su ingreso en 2.2. Millones CUC. Otro proyecto con gran impacto económico contratado es el Sistema de Control de Inventarios, donde el contrato asciende a un monto de 295.322,00 CUC.

Adicionalmente existen como proyectos potenciales nuevos contratos que están en estado de negociación: Sistema de Gestión de Medicamentos (Angola) y Propuesta ERP para empresa Guardián del Alba (Venezuela) con cifras similares de ingreso.

Si se tiene en cuenta que la base arquitectónica y funcional de estas soluciones son los componentes obtenidos y construidos en los proyectos desarrollados dentro del sistema Cedrux y el costo del desarrollo de todos estos proyectos (incluida su implementación) asciende a 920. 970,00 CUP, que representa un 1,4 % del total de ingresos por negocios concertados, se puede arribar a la conclusión que desde el primer negocio contratado se recuperan todos los gastos y se obtienen ganancias.

A estos resultados es válido sumarle los valores por concepto de ahorro de importaciones y pagos de licencia de software similares, dado con el próximo proceso de despliegue de la solución por las entidades del país.

3.5 Conclusiones Parciales

El análisis de reutilización, el experimento realizado y el impacto descrito permiten arribar a:

- El uso de la presente propuesta metodológica en la obtención de los componentes del sistema, permite mediante los índices de reutilización alcanzados, disminuir los tiempos de desarrollo de los proyectos del programa ERP-Cuba.

- El análisis de disminución de los tiempos desarrollo en los proyectos de muestra, brindaron el número de 462 días ahorrado por concepto de reutilización.
- Las pruebas estadísticas utilizadas permitieron constatar del impacto positivo experimentado en las variaciones ocurridas en los proyectos pertenecientes a la población experimental sin afectación negativa en proyectos tomados como muestra de control.
- El análisis realizado del impacto de la solución, demuestra el progresivo cumplimento con las premisas de la dirección del país que dieron inicio al programa ERP-Cuba, así como el comienzo de la actividad de exportación de las soluciones que repercuten directamente en un aumento económico de las ganancias para el país.

Conclusiones

La presente investigación permite arribar a las siguientes conclusiones:

- La definición del marco teórico de la investigación, permitió conocer los principales elementos vinculados con los modelos, métodos y estrategias de desarrollo de arquitectura de software, se realizó un estudio sobre la aplicabilidad de estos modelos y corrientes, en la construcción de la arquitectura de sistema de los proyectos que conforman el programa ERP-Cuba, en este sentido resulta de suma importancia la conclusión de adoptar un estilo orientado a componentes para el desarrollo de la arquitectura de sistema.
- El análisis sobre los principales métodos de desarrollo orientado a la reutilización de activos y más específicamente orientados a la reutilización de componentes, constataron la necesidad del desarrollo de una propuesta metodológica para la obtención de los componentes del sistema que disminuyera los tiempos de desarrollo de los proyectos.
- La propuesta metodológica realizada basa sus principios en la reutilización de los componentes desarrollados y el bajo acoplamiento logrado entre sus componentes, lograda mediante tres elementos fundamentales, una taxonomía de componentes ajustadas a las características del tipo de sistema ERP, una guía de actividades para la obtención de los componentes que conformaran las soluciones a desarrollar y un conjunto de artefactos para la especificación de los componentes obtenidos.
- El análisis realizado sobre los índices de reutilización con los que cuentan los proyectos del programa ERP-Cuba permite concluir la disminución de los tiempos de desarrollo de los proyectos en 462 días.

Recomendaciones

Con el objetivo de dar continuidad a la presente investigación se recomienda:

- Profundizar en los temas referentes a la ingeniería de dominios como paso previo a la definición de los componentes de sistema, con el objetivo de elevar la precisión en las actividades referentes a la obtención de los componentes y aumentar los índices de reutilización en los proyectos.
- Profundizar en los temas referentes a los repositorios de componentes, con el objetivo de agilizar los procesos de reutilización y elevar la calidad de dicha actividad.
- Incorporar a la investigación métrica de estimación de tiempo de desarrollo, teniendo en cuenta los índices de reutilización alcanzados por los proyectos en cada momento, con el fin de elevar la precisión de las actividades de planificación de los proyectos.

Referencias

1. **Maite Sosa, Vladimir González Guerra.** *Sistema de Gestión de Tesis Facultad2.* La Habana, Cuba : Universidad de las Ciencias Informáticas, 2009.

2. **Dr. José Carlos del Toro, Jefe del programa ERP-Cuba, director de política contable.** *Mesa redonda Sistema de gestión de Entidades. Programa ERP Cubano. Producto Cedrux.* III taller internacional de administración financiera. Salón de convenciones. Hotel Habana Libre : GECYT. Gestión del conocimiento y la tecnología, Septiembre 2009.

3. **López Lemus, Yasmary.** *Diseño de un método de selección y evaluación de entidades para el proceso de pruebas piloto de Sistemas Integrales de Gestión.* La Habana, Cuba : CEIGE, Universidad de las Ciencias Informáticas, 2009.

4. **Ríos, Dr.C José Carlos de Toro.** *Documento de Visión del programa ERP-Cuba.* La Habana, Cuba : CEIGE, Universidad de las Ciencias Informáticas, 2007.

5. **Pérez, Msc. Yadenis Piñero.** *Metodología para la gestión de contraración en proyectos.* La Habana, Cuba : CEIGE, Universidad de las Ciencias Informáticas, 2007.

6. **Ing. Dorisbel Muro, Ing. René Lazo Ochoa, Ing. Cesar Lage.** *Reporte de la disciplina de la arquitectura del software desde el enfoque del Rol. Una propuesta de epecialización jerárquica centrada en la responsabilidades y competencias.* UCI, Ciudad de la Habana : s.n., 2009.

7. **Garlan, Mary Shaw y David.** *Software Architecture: Perspectives on an emerging discipline.* Upper Saddle Rive, Prentice Hal. : s.n., 1996.

8. **Electrónicos, Instituto de Ingenieros Electricistas y Electrónicos.** *IEEE Std 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000.

9. **Billy Reynoso, Carlos.** *Introducción a la Arquitectura de Software.* Buenos Aires : Microsoft Press : s.n., 2004.

10. **Osvaldo Díaz Verdecia, Virgen Damaris Quevedo Campins.** *Guía práctica de Arquitectura de Software.* Ciudad de la Habana , Cuba : s.n., 2009.

11. **Corporation, Microsoft.** *MSDN Library para Visual Studio .* [digital] s.l. : Microsoft Corporation, 2008.

12. **Requena, Martín Luis López.** *Microsoft Solutions Framework.* Málaga, España : Microsoft Certified, agosto, 2006.

13. **Marquina, Ernesto y Parra, Jose David.** *Guía de Patrones, Prácticas y Arquitectura .NET.* 2008, febrero. EEUU.

14. **Lars Dragheim Olsen, Michael Fruergaard Pontoppidan, Hans Jorgen Skovgaard, Tomasz Kaminski,.** *Inside Microsoft Dynamics AX.* s.l. : Microsoft Press ©2009 , 2009. 0735626456 9780735626454 .

15. **Javeir Heredia Ruiz, Lilian Alvarez Almanza, Naryana Linares Pons.** *Tendencias en los desarrollos centrados en arquitectura, MDA y la arquitectura de software del centro CEIGE.* La Habana, Cuba : UCIENCIA 2011. (Evento virtual), 2011.

16. **Marta Zorrilla, Belen Vela, Esperanza Marcos.** *Una aproximación dirigida por modelos, para diseñar y construir esquema xml: Un caso de estudio.* Madrid, España : Universidad Rey Juan Carlos., noviembre 2007.

17. **Bustelo, Begoña Cristina Pelayo Garcia.** *Talisman. Desarrollo ágil de software con arquitectura dirigida por modelos.* Universidad de Oviedo (España) : Tesis, 2007.

18. **CARNEGIE-MELLON UNIVERSITY PITTSBURGH PA SOFTWARE ENGINEERING INST.** *A Practical Example of Applying Attribute-Driven Design (ADD), Version 2.0.* EEUU : s.n., febreo 2007. ADA468604.

19. **Daniel Perovich, Maria Cecilia Bastarrica, Cristian Rojas.** *Model-Driven approach to Software Architecture design.* IEEE Computer Society Washington, DC, USA ©2009 : s.n., 2009. 978-1-4244-3726-9.

20. **Olaf Zimmermann, Thomas Gschwind, Jochen Küster, Frank Leymann, Nelly Schuster7.** *Reusable Architectural Decision Models for Enterprise Application Development* . s.l. : Lecture Notes in Computer Science,Volume 4880/2007, 15-32, 2007. DOI: 10.1007/978-3-540-77619-2_2.

21. **Aruquipa Chambi Marcelo G., Márquez Granado Edwin P.** *Desarrollo de Software Basado en Componentes.* Universidad Mayor de San Andrés – La Paz Bolivia : s.n., agosto 2007.

22. **Martínez, L.** *"Un Modelo de Mediación para el desarrollo de software basado en Componentes COTS.".* 04210, Ctra Sacramento s/n, Almería, España : Departamento de Lenguajes y Computación, Universidad de Almería, 2003.

23. **SEI.** Software Engennering Institute, Process Area Components . [En línea] SEI, 2008. [Citado el: 15 de marzo de 2011.] http://www.sei.cmu.edu/.

24. **C.Wallnau, A. W. Brown and K.** *The current state of CBSE.* s.l. : IEEE Software, 15(5):37-46, 1998.

25. **Szyperski., C.** Component software: Beyond object-oriented programming. Noviembre 2002.

26. **Montilva, J.** "Desarrollo de Software Basado en Componentes. 2005.

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

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

29. **Object Management Group.** Common object request broker architecture: Core specification.

[En línea] diciembre de 2002. http://www.omg.org/docs/for.

30. *T. Digre. Business object component architecture.* **IEEE Software.** s.l. : Journal, EEE Computer Society Press Los Alamitos, CA, USA , septiembre 1998, Vol. Volume 15 Issue 5.

31. **Addison-Wesley.** *I. Sommerville. Software engineering.* s.l. : ISBN: 020139815X, agosto 2000. Pub Co, 6ta edición..

32. **Sametinger., J.** *Software engineering with reusable components.* Austria : Institut fur Wirtschaftsinformatik , agosto 1997. ISBN 3-540-62695-6.

33. **J. Sodhi, y P. Sodhi.** *Software reuse: Domain analysis and design process.* NC USA : Software Development, 1999. ISBN-10: 0070579237 --- ISBN-13: 978-0070579231.

34. **Silva, Armando Cabrera.** *Ingenería del software, guía didáctica.* UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA, Ecuador : s.n., ABRIL-AGOSTO 2009. 18701.

35. **Lidia Fuentes, José M. Troya y Antonio Vallecillo.** *Desarrollo de Software Basado en Componentes.* Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga. : s.n., septiembre 2004. 29071 Málaga, Spain.

36. **Argudo Vásconez, Joaquín Andrés.** *Estudio de las Metodologías de desarrollo de Software Libre y su aplicación en un caso práctico.* oct 2010.

37. **Jonás A. Montilva, Judith Barrios, Vannesa Hammar.** *Aspectos metodológicos del desarrollo de componentes reutilizables.* Guadalajara Mexico : s.n., Noviembre 2004.

38. **Hamar, Vannesa.** *Apectos metodológicos del desarrollo y reutilización de componentes de software.* Merida, Venezuela : s.n., noviembre 2003.

39. **Metzger, A., y otros, y otros.** *Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis.* Alemania : s.n., octubre 2007. 978-0-7695-2935-6 .

40. **Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson.** *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* s.l. : CMU/SEI-90-TR-21, noviembre 1990. ESD-90-TR-222.

41. **Holibaugh, R.** *Join Integrated Avionics Working Group (JIAWG) Object-Oriented Domain Analys Method (JODA).* s.l. : Carnegie Mellon University, 1992. SEI-92-SR-3.

42. **J, Odell J.** *A Primer to Method Enginnering INFOSYS: The Electronic newsletter for information system.* 1996.

43. **Ing. Dorisbel Muro Fumero1, Ing. René Lazo Ochoa1, Ing. Cesar Lage Codorniu1.** *Propuestas de modelos de desarrollo centrado en la arquitectura, para ambientes de producción industrial.* Ciudad de la Habana, Cuba : Uciencia 2010., 2010.

44. **Osmar, Leyet.** *Documento de descripción de la arquitectura de software.* La Habana, Cuba :

Universidad de las Ciencias Informáticas, 2010.

45. **Lage, Cesar.** *Solución arquitectónica de la Configuración General del sistema para la parametrización de negocio de Cedrux.* La Habana : Universidad de las Ciencias Informáticas, 2011.

46. **Paul Clements, Rob Wojcik, Felix Bachmann, Len Bass, Paulo Merson, Robert Nord, Bill Wood.** *Attribute-Driven Design (ADD).* Pittsburgh, EEUU : s.n., noviembre 2006. 01731-2100.

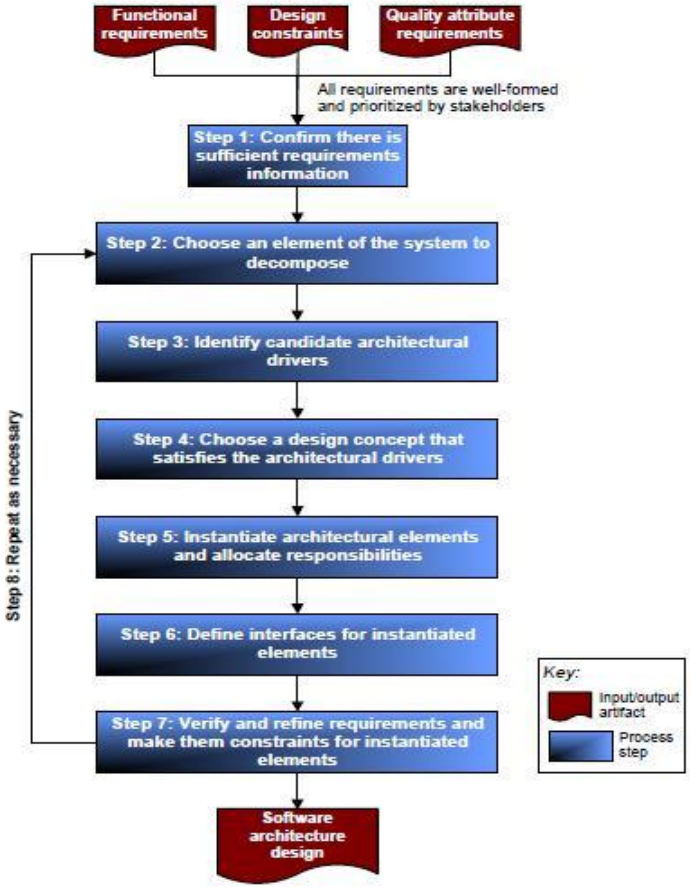
47. **Pablo, Fillotrani.** *Calidad en el Desarrollo de Software, Modelos de Calidad de Software. Departamento de Ciencias e Ingeniería de la Computación.* Universidad Nacional del Sur : s.n., 2007.

48. **Len Bass, Paul Clements, Rick Kazman.** *Software Architecture in Practice, Second Edition. Adison Wesley.* 2003. ISBN: 0-321-15495-9.

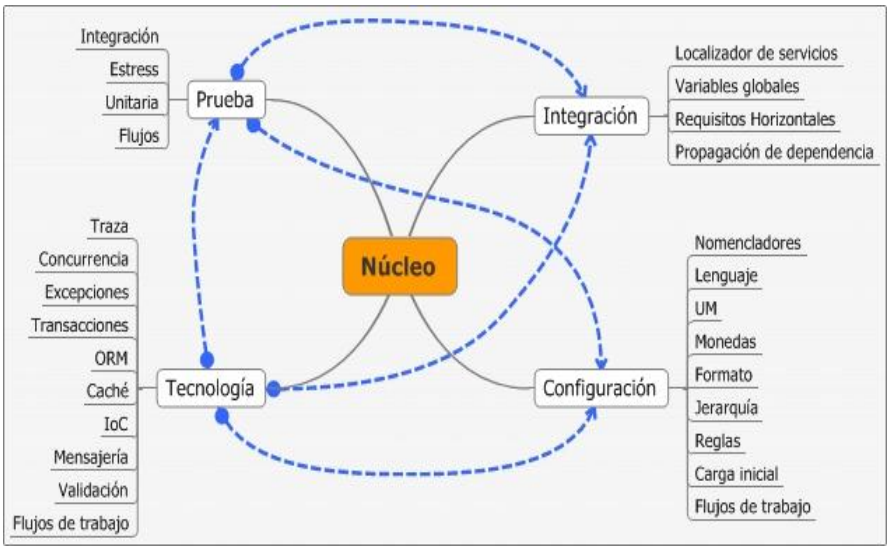
49. **MSc. Yadenis Piñero Pérez, Ing. Johanny Rivera López.** *Cedrux, Experiencias del pilotaje.* La Habana, Cuba : Revista Cubana de Gestión Empresarial, Nueva Empresa, 2011. 1682-2455.

50. **Firesmith, Donald.** *QUASAR: A method for the Quality Assesment of Software-Intensive Systems Architectures.* s.l. : Carniege Mellon University, Software Engineering Institute, 2006.

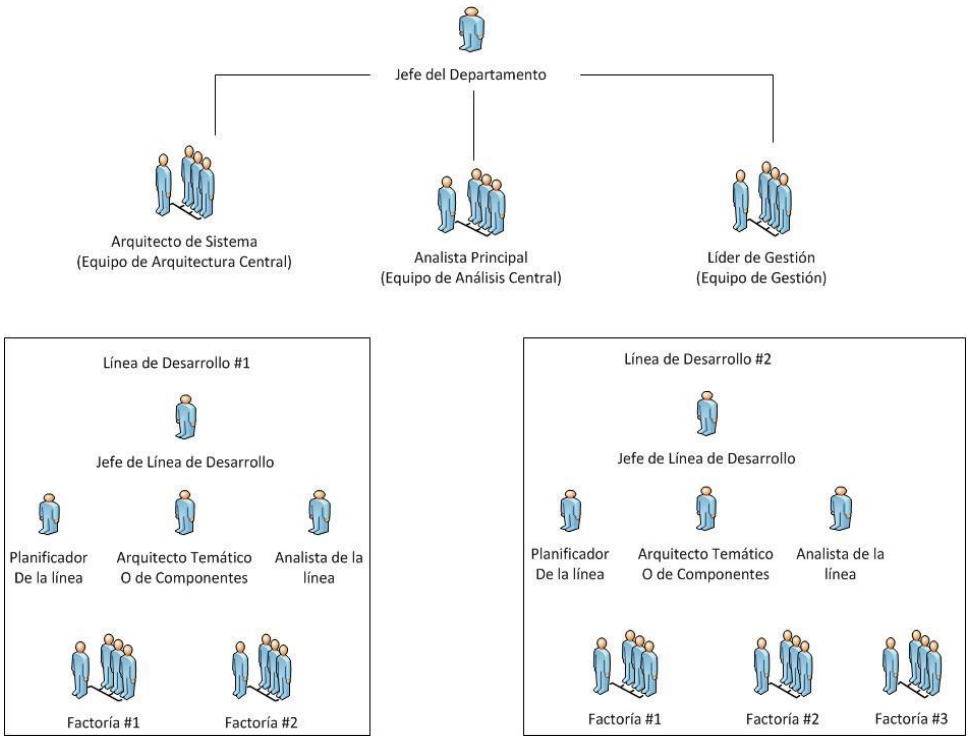
Anexos



Anexo 1 Fases del modelado arquitectónico propuesto por el ADD del SEI.



Anexo 2 Tipos de componentes.



Anexo 3 Estructura organizativa del equipo.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/SistemaIntegralGestion/MATRIZ ANÁLISIS DE ATRIBUTOS DE CALIDAD - RESTRICCIONES DEL DISEÑO.xlsx

Anexo 4 Documento Matriz análisis de atributos de calidad-restricciones del diseño.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/SistemaIntegralGestion/EspecificaciónComponentesContenedores.xlsx.

Anexo 5 Documento Especificación de componentes contenedores.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/SistemaIntegralGestion/EspecificacionComponenteContenedor.xlsx

Anexo 6 Documento Especificación de las propiedades del componente contenedor.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/SistemaIntegralGestion/MatrizGeneraldeIntegracion.xlsx

Anexo 7 Documento Matriz general de integración.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/Subsistemas/EspecificacionComponente.xlsx

Anexo 8 Documento Especificación de componente.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/Subsistemas/EspecificacionPropiedadesComponente.xlsx

Anexo 9 Documento Especificación de las propiedades de los componentes.

https://repositorio.ceige.prod.uci.cu/svn/repo/CSGPROD/Expediente_Proyectos/Disciplinas%20ingenieriles/Artefactos - Arquitectura (Vista Sistema)/Subsistemas/MatrizIntegracionComponentesNivel1.xlsx

Anexo 10 Documento Matriz de integración de componentes nivel1.