UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 4 y Laboratorio de Soluciones e Investigaciones Avanzadas en Gestión de Proyectos

Propuesta de solución de la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Melvin Delgado Beretervide

Tutor: Ing. Yusdel Meriño Almaguer
Cotutor: Lic. Dania Domínguez Álvarez

La Habana, Junio 2012
"Año 54 de la Revolución"

"Si fuéramos capaces de unirnos... Qué hermoso y qué cercano sería el futuro!"

Ernesto Ché Guevara

Declaración de autoría

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con él. Para que así conste firmo la presente a los __ días del mes de __ del año __.

Melvin Delgado Beretervide

_____

Firma del autor

Ing.Yusdel Meriño Almaguer

_____

Firma del tutor

Agradecimientos

A mi tutor Yusdel Meriño Almaguer, que sin su ayuda y sus consejos me hubiera sido imposible la confección de este documento, por su paciencia a la hora de revisarme la tesis y por los días enteros que tuvo que dedicarle en su tiempo extra.

A mi jefe de tribunal Félix Noel Abelardo Santana, que me ayudó incondicionalmente, cada vez que necesité de él.

A mi oponente Lizardo Ramírez Tabuada por su aporte en la confección de mi tesis y sus consejos.

Gracias por todo.

A mis padres por haberme apoyado y haber confiado siempre en mí.

A mi hermana a mis abuelos y a toda mi familia en general por haberme apoyado en todo momento.

A mi novia por ayudarme y apoyarme en todo momento, por alentarme y darme fuerzas cuando pensaba que no se podía.

Gracias por todo BB.

A Leannet porque sin su compañía y ayuda me hubiera sido imposible llegar a este 5to año de la carrera.

Muchas gracias.

A la Chacha mía por preocuparse por mí siempre y ser mi 2da madre y por acogerme como si fuera su hijo.

Gracias por todo.

A mis amigos del IPVCE: Omarito, Greiyam, Salgado, Double que siempre se preocuparon por mí.

A mis amistades, esas que son tantas y no puedo mencionar a algunos y a otros no. A ustedes que en algún momento me ayudaron con un consejo o alguna idea.

A mis compañeros de aula que de una forma u otra forman parte de mi vida aquí en la Universidad.

A nuestros profesores, que sin su enseñanza nuestros sueños no se hubieran hecho realidad.

Y a todos aquellos que de una forma u otra contribuyeron a la realización de este trabajo.

Melvin

Dedicatoria

A mis padres, Olga Lidia Beretervide Rodríguez y Ramón Delgado Acosta, a quienes les estaré eternamente agradecido por darme esta oportunidad de estudiar y saber siempre que cuento con su apoyo desinteresado, en los momentos buenos y sobre todo, en los malos, que siempre han confiado en mí y no los defraudaré. A ustedes les digo que me esforcé todo lo que pude y que ya casi tengo un gran sueño realizado, el tener un título universitario, en una carrera que me gusta y se además, que este es un sueño de ustedes también.

A mis abuelos quienes quiero y aprecio mucho ya que siempre han estado presentes con su ejemplo y me han dado consejos en la vida a través de sus cuentos y sus regaños. Para ustedes también va dedicada esta tesis, y aunque quizás no puedan leerla, sepan que los llevo en el corazón y su ejemplo está en mi recuerdo cada vez que obtengo un logro.

A mi hermana Lisbet Delgado Beretervide, la cual siempre me dio su apoyo. A ti te digo que no importa cuántas veces la vida te haga caer, ella te está poniendo a prueba, lo realmente importante es levantarse y seguir con más bríos y fuerzas que antes.

A mi novia Lidis Corcho Gómez que con mucho esfuerzo y paciencia me ayudo en este logro. A ti digo que caíste del cielo para mi alegría, fueron llantos de felicidad y de gozo, el mundo se tiñó de tus ojos y sucumbí ante tu dulzura… gracias por existir.

A mi 'bichito' que siempre estuvo presente desde mi 1er año en la escuela hasta este último. A ti te digo que doy gracias al cielo por haberte conocido, que aunque no estemos cerca sepas que eres importante para mí. Gracias a ti aprendí grandes cosas de la vida.

A mi 'Chacha' que me acogió como un hijo desde antes de conocerme. Para ti también va dedicado este logro, decirte que te considero como una madre y que siempre va a ser así.

A mis primos en general, gracias por apoyarme siempre y por ser parte de mi vida. A los de parte de madre por ayudarme siempre y a los de parte de padre por ser siempre un ejemplo a seguir para mí.

Y a todas aquellas personas que han confiado en mí y en algún que otro momento me han tendido su mano solidaria, o me han dado algún consejo.

A todos muchas gracias

Melvin.

Resumen

En la actualidad las instituciones cubanas han incrementado el uso de aplicaciones informáticas, con el objetivo, entre otros, de tener un control eficiente sobre los recursos que se manejan. Entre las aplicaciones informáticas puestas en práctica se encuentran los sistemas de gestión de proyectos, que son softwares para integrar y automatizar todos los procesos que se llevan a cabo en las instituciones. Actualmente en la Universidad de la Ciencias Informáticas (UCI) se desarrolla el paquete de Gestión de Proyectos GESPRO, donde se tiene como propósito, definir e implantar la Arquitectura del Sistema en aras de mejorar la calidad de su oferta. Con el presente trabajo se pretende proponer una arquitectura para la Vista de Sistema de la Plataforma GESPRO en su versión 12.05.

La presente investigación tiene como objetivo proponer una solución a la Vista de Sistema de la Arquitectura de GESPRO v.12.05. Para el desarrollo de la propuesta se trabajó con los requisitos funcionales del sistema, y se obtuvo como resultado la priorización de los mismos atendiendo a su relevancia arquitectónica. Se logró agrupar semánticamente los componentes candidatos de la arquitectura de sistema, identificándose los paquetes principales y las dependencias entre ellos. Se culminó la propuesta con la descripción de los escenarios y patrones de solución, la misma fue validada mediante el método de pre-experimento.

Abstract

Today the Cuban institutions have increased the use of computer applications, with the aim, among others, to have efficient control over resources are managed. Among the applications are implemented management systems projects, which are software to integrate and automate all processes carried out in institutions. Currently at the University of Informatics Sciences (UCI) develops the package GESPRO Project Management, which aims to define and implement System Architecture in order to improve quality of their offer. The present study sought to propose architecture for the System View of the GESPRO Platform in the version 12.05.

After culminated the application a proposed solution to the System View Architecture GESPRO 12.05. The same includes actions to characterize the product depending on the view. For the development of the proposal is worked with the functional requirements of the system, and resulted in the prioritization of them according to their relevance architecture. It was possible to group semantically candidate components of the system architecture, identifying core packages and dependencies between them. It culminated in the description of the scenarios and patterns of settlement.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

En las últimas décadas se ha producido un vertiginoso cambio científico-tecnológico, que ha dado lugar a la sociedad de la información. La aparición de nuevas tecnologías como el ordenador, el teléfono móvil e Internet, ha producido una revolución social; principalmente, porque ofrecen posibilidades de comunicación e información con el mundo. La Industria de Software a su vez se ha convertido en una de las más poderosas y crecientes del momento. Cada día son más las empresas que se suman motivadas por la competencia y el auge de productos de alta calidad. El desarrollo del software se ha convertido en una necesidad para la sociedad.

Cuba ha decido no quedarse al margen de la revolución tecnológica y la automatización de procesos, por lo que se encuentra inmersa en la labor de informatización de la sociedad. Un paso decisivo en este aspecto fue la creación de la Universidad de las Ciencias Informáticas (UCI), centro de altos estudios basado en el concepto de universidad productiva, donde convergen las necesidades de formación y de producción de soluciones informáticas con el objetivo de lograr recursos humanos formados, desarrollar la informatización cubana e ingresar divisas por concepto de exportación.

La gestión de proyectos es un tema que se trata en el centro. La inclusión de herramientas para la ayuda a la toma de decisiones facilita significativamente el control y seguimiento de los proyectos y de los recursos materiales y humanos asociados a los mismos. Es por ello que surge en la Universidad, el Sistema GESPRO, un paquete que incluye módulos que permiten entre otras funcionalidades: la gestión de portafolios de proyectos, la gestión de alcance, la gestión de tiempo, la gestión de riesgos, la gestión de comunicaciones, la gestión de la calidad, la gestión de recursos humanos, monitoreo y control de la plataforma, el control de versiones y la gestión documental. Es de gran importancia que se tome en cuenta la arquitectura de esta plataforma.

El desarrollo de la arquitectura de los sistemas es la columna vertebral de la integración, del mantenimiento y el desarrollo de los mismos. La definición y formalización de la arquitectura de los sistemas constituye entonces un elemento importante que permite a las organizaciones tener adecuadamente descritos los sistemas, disminuyendo el riesgo de pérdida de información por concepto de falta de personal, con la implementación y adecuada formalización de las arquitecturas de los sistemas se logra aumentar el conocimiento organizacional y se mitigan los riesgos por pérdida de recursos humanos. En este sentido además como parte de los servicios de comercialización de GESPRO se proponen servicios de transferencias que incluyen la entrega de documentación de la arquitectura del sistema y que implican un impacto económico importante para la universidad y el país.

Se conoce que en la versión anterior de GESPRO (GESPRO v 1.0) existían una serie de dificultades que atentaban contra la calidad del trabajo que se desarrolla en la entidad. Al realizar un análisis de la situación se detectaron los siguientes problemas:

- No se realizaba un levantamiento formal de los requisitos del sistema, incluyendo los requisitos funcionales que poseen un mayor impacto en la arquitectura del sistema y constituyen la columna vertebral de la solución.
- No se priorizaban ni agrupaban los requisitos con respecto a su relevancia arquitectónica.
- No se establecían los diferentes escenarios de la Plataforma.
- Existía una mala planificación de la calidad al documentar de forma digital los requisitos del sistema.

Se necesita entonces que la nueva versión de GESPRO, en la Vista de Sistema, pueda dar solución a estas dificultades a través del desarrollo de un grupo de acciones que permitan alcanzar una adecuada estructuración del sistema.

Luego de analizar la problemática anterior, se tiene como **problema a resolver:** la insuficiencia en la Gestión de los Requisitos Funcionales de la Arquitectura del Sistema GESPRO 12.05, afecta la planificación de la calidad enfocado a la documentación digital de los requisitos.

Es por ello que se define como **objeto de estudio** la Arquitectura del Sistema GESPRO.

Para resolver el problema planteado el **objetivo general** que se persigue con este trabajo es: Desarrollar la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05 y el **campo de acción** sería**:** La Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05.

Para darle cumplimiento al objetivo general se definen los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Definir la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05.
- ✓ Implantar la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05.
- ✓ Validar los resultados esperados.

El **tipo de investigación** para poder resolver el problema que se muestra será la **explicativa.**

Se plantea como **hipótesis** lo siguiente**:** si se define e implanta la Vista de Sistema de la Arquitectura de la Plataforma GESPRO se alcanzarán mejoras en la planificación de la calidad enfocado a los requisitos del Sistema.

La **variable dependiente** es**:** la planificación de la calidad enfocada a la documentación digital de los

requisitos.

La **población** a estudiar será: los centros de la red de producción de la UCI.

La **muestra** es: la red de centros de la sede central.

Los métodos teóricos:

1. El método **Histórico-Lógico:** se utilizó en el análisis de las tendencias actuales de los modelos arquitectónicos.
2. El método **Análisis y Síntesis:** se utilizó en el trabajo con la información y el arribo a las conclusiones de la investigación.

Los métodos empíricos:

1. El método de la **Entrevista**: se utilizó este método en la investigación debido a que gran parte de la información está en manos de los especialistas.
2. El método de la **Observación**: se utilizó este método para la investigación dado que existía una versión anterior y se pueden constatar los errores de la misma.

Con la definición e implantación de la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05, se espera como **aporte práctico** optimizar el proceso de gestión de los requisitos para los proyectos, mejorándose así la planificación de la calidad de la documentación digital de los requisitos funcionales.
El **diseño de investigación** es el **pre-experimento** de pre y post prueba con un solo grupo, en la que hay al menos un punto de referencia inicial. La validez interna puede ser afectada fácilmente por la historia, la maduración, la elección de un grupo atípico, la regresión y las interacciones.

Este trabajo se divide en tres capítulos para darle solución a la investigación.

Capítulo 1: Fundamentación Teórica: En este capítulo se hace un estudio del estado del arte de la Arquitectura de Software. Se definen conceptos de importancia para los términos de Arquitectura de Software y Vistas arquitectónicas. Se abordan temas referentes a los diferentes modelos arquitectónicos que están vigentes actualmente. Se realiza un estudio de los elementos a tener en cuenta para desarrollar la propuesta de solución de la Vista de Sistema de la plataforma GESPRO.

Capítulo 2: Propuesta de Solución: En este capítulo se realiza la propuesta de solución para el problema de la investigación, en el se identifican los requisitos funcionales priorizados del sistema, después se le halla la relevancia arquitectónica a cada uno de ellos, para luego agruparlos semánticamente en paquetes o componentes y culmina la propuesta de solución con la representación de la dependencia entre paquetes o componentes y la determinación de los escenarios presentes en la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05.

Capítulo 3: Escenarios de solución. Validación de la propuesta: En este capítulo se hace un estudio de diferentes métodos de evaluación para luego aplicarlos a la propuesta de solución y ver si es factible su aplicación y ver los resultados obtenidos.

Capítulo 1.Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realiza la fundamentación teórica de la investigación, haciendo un análisis del estado del arte de la Arquitectura de Software. Se exponen los conceptos más importantes relacionados con esta disciplina, y se detallan los tipos de arquitectura de software más universales. Se hace un análisis además sobre las principales escuelas o corrientes reconocidas para esta rama de la informática. Se conceptualiza el término Vista Arquitectónica, y se abordan temas de importancia referentes a los diferentes modelos arquitectónicos utilizados en el mundo y en la Universidad. Se concluye con la presentación de los elementos que componen la Vista de Sistema, principal objetivo de la presente investigación.

1.2 Definiciones de la Arquitectura de Software

La arquitectura de software brinda una estructura del producto, y es fundamental en el proceso de vida del software, consiste en un conjunto de estilos, vistas, patrones y abstracciones coherentes, entre otros términos que son los que en realidad dirigen la construcción del sistema.
La arquitectura se basa específicamente en una serie de objetivos y restricciones funcionales y no funcionales que le dan al sistema toda la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas.

Pressman plantea:
“La arquitectura no es el software operacional, más bien es la representación que capacita al ingeniero del software para: analizar la efectividad del diseño para la consecuencia de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil y reducir los riesgos asociados a la construcción del software” (1).

Bas y otros plantearon:
“La arquitectura del software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los elementos del software, las propiedades de esos elementos del software visibles externamente, y las relaciones entre ellos” (2).

Garlan y otros por su parte refieren:
“...es el nivel del diseño del software donde se definen la estructura y propiedades globales del sistema”.
“...se centra en aquellos aspectos del diseño y desarrollo que no pueden tratarse de forma adecuada dentro

de los módulos que forman el sistema” (3).
“Más allá de los algoritmos y estructuras de datos de la computación; el diseño y la especificación de la estructura general del sistema emergen como una clase nueva de problema. Los aspectos estructurales incluyen la estructura global de control y la organización general; protocolos de comunicación, sincronización y acceso de datos; asignación de funciones para diseñar elementos; distribución física, composición de elementos de diseño; ajuste y rendimiento; y selección entre otras alternativas de diseño” (4).

Por otro lado, Philippe Kruchten en el *Rational Unified Process* (RUP) plantea:
“La arquitectura de software representa la estructura o las estructuras del sistema, que consta de componentes de software, las propiedades visibles externamente y las relaciones entre ellas” (5).

Por tanto, se puede afirmar que la arquitectura del software brinda una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. Es una vista estructural de alto nivel; tiene un estilo o combinación de estilos para una solución, está estrechamente relacionada al trabajo en equipo y la correcta planificación y costo del software y es vital para el éxito o fracaso de un proyecto. Existen muchas otras definiciones de Arquitectura del Software, pero en un sentido amplio el autor del presente trabajo está de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación.

1.3 Escuelas de la Arquitectura

Aunque no existe hoy en día nada que avale explícitamente la existencia de escuelas de Arquitectura de Software, ni una referencia que analice las particularidades de cada una, en la actualidad se pueden distinguir a grandes rasgos unas seis corrientes. El pertenecer a una de estas corrientes no representa algo determinista, ya que en distintos momentos algunos practicantes de la Arquitectura de Software realizan diferentes trabajos operando en marcos distintos, o porque simplemente cambian de concepción.

Se pueden distinguir a grandes rasgos seis corrientes principales de Arquitectura de Software:

1.3.1 Arquitectura como etapa de ingeniería y diseño orientada a objetos:

Grady Booch plantea que la Arquitectura de Software es “la estructura lógica y física de un sistema, forjada por todas las decisiones estratégicas y tácticas que se aplican durante el desarrollo”.(13) Otras definiciones revelan que la Arquitectura de Software, 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 Kruchten (11).

1.3.2 Estructuralismo arquitectónico radical:

Se trata de un desprendimiento de la corriente anterior, mayoritariamente europeo, que asume una actitud más confortativa con el mundo UML (*Unified Modeling Language*). En el seno de este movimiento hay al menos dos tendencias, una que excluye de plano la relevancia del modelado orientado a objetos para la Arquitectura de Software y otra que procura definir nuevos meta-modelos y estereotipos de UML como correctivos de la situación (13).

1.3.3 Arquitectura basada en patrones:

Si bien reconoce la importancia de un modelo emanado del diseño orientado a objetos, no se encuentra tan rígidamente vinculada a UML en el modelado, ni a CMM en la metodología. El texto sobre patrones que esta variante reconoce como referencia es la serie POSA (*Pattern-Oriented Software Architecture*) de Buschmann y otros, y secundariamente el texto de la Banda de los Cuatro. El diseño consiste en identificar y articular patrones preexistentes, que se definen en forma parecida a los estilos de arquitectura (8).

1.3.4 Arquitectura procesual:

Desde comienzos del siglo XXI, con centro en el *Software Engineering Institute* (SEI) y con participación de algunos de los arquitectos de Carnegie Mellon de la primera generación y muchos nombres nuevos de la segunda: Rick Kazman, Len Bass, Paul Clements, entre otros. Intenta establecer modelos de ciclo de vida y técnicas de elicitación de requerimientos, lluvia de ideas, diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la Arquitectura de Software (9).

1.3.5 Arquitectura basada en escenarios:

Es la corriente más actual. 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. En esta corriente suele utilizarse diagramas de casos de uso UML como herramienta informal u ocasional, dado que los casos de uso son uno de los escenarios posibles. Los casos de uso no están orientados a objeto. Los autores vinculados con esta modalidad han sido, aparte de los codificadores de A*rchitecture Tradeoff Analysis Method (*ATAM), *Cost-*

Benefits Analysis Method (CBAM), *Quality Attribute-Oriented Software Architecture Design Method* (QASAR) y demás métodos del SEI, los arquitectos holandeses de la Universidad Técnica de Eindhoven, de la Universidad Brije, de la Universidad de Groningen y de Philips Research (6).

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

Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon. Se trata también de la visión de la Arquitectura de Software dominante en la academia, y aunque es la que ha hecho el esfuerzo más importante por el reconocimiento de la Arquitectura de Software como disciplina, sus categorías y herramientas son todavía mal conocidas en la práctica industrial. En el interior del movimiento se pueden observar distintas divisiones. Hay una variante informal de “boxología” y una vertiente más formalista, representada por el grupo de Mark Moriconi en el *Stanford Research Institute* (SRI) de Menlo Park. En principio se pueden reconocer tres modalidades en cuanto a la formalización; los más informales utilizan descripciones verbales o diagramas de cajas, los de talante intermedio se sirven de ADLs y los más exigentes usan lenguajes formales de especificación como CHAM y Z. En toda la corriente, el diseño arquitectónico no solo 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, y en general nadie habla de clases o de objetos. Mientras algunos participantes excluyen el modelo de datos de las incumbencias de la Arquitectura de Software (Shaw, Garlan, entre otros), otros insisten en su relevancia (21).

Se llega a la conclusión de en que las corrientes representativas predomina como factor común: las vistas arquitectónicas, que describen características de la arquitectura según el contexto a desarrollar, además de que cada una de las corrientes no integra los elementos teóricos del resto.

1.4 Tipos de Arquitecturas existentes

Son varios los tipos de arquitecturas que existen en la actualidad, seguidamente se muestran algunos de los más importantes:

1.4.1 Arquitectura orientada a objetos:

A este tipo de arquitectura se le conoce de varias maneras, entre ellas: arquitecturas basadas en objetos, abstracción de datos y organización orientada a objetos. Los componentes de esta son los objetos, o más bien instancias de los tipos de datos abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son

responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos (7).

Según Billy Reinoso, si hubiera que resumir las características de las arquitecturas orientadas a objetos, se podría decir que: los componentes se basan en principios orientados a objetos como encapsulamiento, herencia y polimorfismo. Son así mismo las unidades de modelado, diseño e implementación. Los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación (10).

Las interfaces están separadas de las implementaciones. En general la distribución de objetos es transparente, y en el estado de arte de la tecnología apenas importa si los objetos son locales o remotos. El mejor ejemplo de orientación a objetos para sistemas distribuidos es *Common Object Request Broker Architecture* (CORBA), en la cual las interfaces se definen mediante *Interface Description Language* (IDL); un *Object Request Broker* (intermediarios en peticiones a objetos) media las interacciones entre objetos clientes y objetos servidores en ambientes distribuidos (12).

En cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases. Hay muchas variantes; algunos sistemas, por ejemplo, admiten que los objetos sean tareas concurrentes; otros permiten que los objetos posean múltiples interfaces.

Ventajas:

- Se puede modificar la implementación de un objeto sin afectar a sus clientes.
- Un objeto es una entidad reutilizable en el entorno de desarrollo.
- Encapsulamiento.

Desventajas:

- Para poder interactuar con otro objeto a través de una invocación de procedimiento, se debe conocer su identidad.
- Presenta problemas de efectos colaterales en cascada: si A usa B y C también lo usa, el efecto de C sobre B puede afectar a A.
- Granularidad muy fina para sistemas grandes.

1.4.2 Arquitectura en Capas:

Garlan y Shaw la definen como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.

Los conectores se definen mediante los protocolos que determinan las formas de la interacción. Las restricciones topológicas pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma. Es de suponer que si esta exigencia se relaja, el estilo deja de ser puro y pierde algo de su capacidad heurística (7); también se pierde, naturalmente, la posibilidad de reemplazar totalmente una capa sin afectar a

las restantes, disminuye la flexibilidad del conjunto y se complica su mantenimiento. En muchas ocasiones se sacrifica la pureza de la arquitectura en capas precisamente para mejorarla, colocando, por ejemplo: reglas de negocios en los procedimientos almacenados de las bases de datos o articulando instrucciones de consulta en la capa de la interfaz de usuario.

Ventajas:

- Modularidad.
- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Proporciona amplia reutilización.
- Soporta fácilmente la evolución del sistema; los cambios so lo afectan a las capas vecinas. Se pueden cambiar las implementaciones respetando las interfaces con las capas adyacentes.

Desventajas:

- Es difícil razonar a priori sobre la separación en capas requeridas.
- Formatos, protocolos y transportes de la comunicación entre capas suelen ser específicos y propietarios.
- No todos los sistemas pueden estructurarse en capas.

1.4.3 Arquitectura orientada a servicios (SOA):

La arquitectura orientada a servicios *Service Oriented Architecture* con sus siglas en inglés (SOA) está formada por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma y del lenguaje de programación por ejemplo: El *Web Services Description Language* (WSDL). La definición de la interfaz encapsula las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo, como *Java* o .NET.

Con esta arquitectura, se pretende que los componentes de software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio desarrollado en *CSharp* podría ser usado por una aplicación *Java* (14).

Una de las características más relevante de SOA, es que está basada en contratos, donde el proveedor establece las reglas de comunicación, el transporte, y los datos de entrada y salida que serán intercambiados por ambas partes.

Otras características de SOA son:

- Es una arquitectura conceptual.
- Organiza funciones de negocio como servicios ínter operables.
- Permite reutilización de servicios para satisfacer necesidades de negocio.
- Es basada en estándares. Independencia de fabricantes.

- Es una arquitectura en la cual se exponen los procesos de negocio del sistema a construir como servicios independientes de alta cohesión y bajo acoplamiento que encapsulan dichos procesos y pueden ser invocados a través de interfaces bien definidas.

Ventajas:

- Mejora la toma de decisiones.
- Panorámica unificada. Más información con mejor calidad.
- Potencia la relación entre clientes y proveedores.
- Mayor capacidad de respuesta a los clientes.
- Aplicaciones más productivas y flexibles.
- Aplicaciones más seguras y manejables.

Desventajas:

- Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, entre otros. Esto necesariamente implica la utilización de mensajería confiable.
- La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red, configuración, entre otros.
- Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etcétera.

1.4.4 Arquitectura basada en componentes:

Actualmente en el desarrollo de software hay una gran necesidad de hacer uso de la reutilización de partes o módulos de software existentes, que podrían ser utilizados para la generación de nuevas extensiones de las aplicaciones o las aplicaciones completas.

Al ser una arquitectura basada en componentes permite la reutilización de los mismos. Para ello los componentes deben satisfacer como mínimo el siguiente conjunto de características:

- Identificable: un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.
- Accesible sólo a través de su interfaz: el componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- Servicios son invariantes: las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.
- Documentado: un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

Ventajas:

- Reutilización del software. Lleva a alcanzar un mayor nivel de reutilización de software.
- Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Desventajas:

- Si no existen los componentes, hay que desarrollarlos y se puede perder mucho tiempo, por otro lado, estos componentes pueden tener conflictos si de estos sale una nueva versión, por lo que es posible que haya que re-implementarlos.

Es importante mencionar que cada una de las arquitecturas antes descritas es usada para dar solución acorde a las necesidades y servicios que brinde un producto.

1.5 Definiciones de Vistas Arquitectónicas

Los marcos arquitectónicos al igual que las metodologías de modelado de los organismos acostumbran a ordenar las diferentes perspectivas de una arquitectura en términos de vistas.

Desde los inicios de la arquitectura de software se planteó la necesidad de contar con varias vistas para separar a los diferentes elementos del software.

'…el arquitecto del software necesita un numero de vistas diferentes de la arquitectura de software para varios usos y usuarios…son requeridas para enfatizar y entender diferentes aspectos de la arquitectura…' (15).

Existen diferentes definiciones para el término Vista:

"un subconjunto resultante de practicar una selección o abstracción sobre una realidad, desde un punto de vista determinado" (15).

De acuerdo a la definición del estándar 1471 de la IEEE, es lo siguiente:

"Una vista es una representación de un sistema completo desde la perspectiva de un conjunto de *concerns* relacionados" (16).

En conclusión una vista no es más que una temática desde la que se enfoca el proyecto, ya sea desde el punto de vista de los datos, integración, entre otros y que hace posible enfrentar el diseño de la arquitectura desde diversas perspectivas reduciendo la complejidad presente en su desarrollo.

Las vistas arquitectónicas constituyen las estructuras fundamentales de la arquitectura. El diseño de las vistas arquitectónicas implica establecer una sincronización entre sus elementos para conformar una arquitectura integrada, y de poder ser capaz de mantener la consistencia entre sus elementos. Cada vista agrupa a elementos que pertenecen a diferentes intereses, además de esto se llegan a establecer relaciones de correspondencia entre los elementos de distintas vistas, puesto que al formar parte de un mismo sistema de alguna manera se llegan a relacionar. Estas relaciones de correspondencia constituyen el adhesivo que mantiene el enlace a las vistas que integran a la arquitectura del software.

Para separar las vistas se debe tener en cuenta todo el trabajo que se lleva a cabo a la hora de estructurar una arquitectura, tratando de no dejar ningún contenido fuera de alguna de ellas. Esto como es lógico no es una labor nada sencilla, y mientras más se crece en los conocimientos arquitectónicos más complejo y engorroso se vuelve, generando muchas dudas sobre dónde va un contenido, si es suficiente con integrarlo a alguna vista ya propuesta o si es indispensable crear una nueva vista solo para él.

1.6 Modelos Arquitectónicos Basados en Vistas

A continuación se describen algunos de los modelos arquitectónicos que se rigen por la idea de las principales escuelas o tendencias de la arquitectura

RUP propone el modelo de Arquitectura 4+1 Vistas de Kruchten (5):

- Vista lógica: requerimientos de comportamiento, mecanismos comunes de diseño (basada en diagramas de objetos y clases).
- Vista de procesos: distribución, integridad, tolerancia a fallas (basada en describir una red lógica de programas que se comunican).
- Vista de desarrollo o implementación: rehúso, portabilidad, asignación de requerimientos y trabajo de equipos. Organización del software en el ambiente de desarrollo.
- Vista física: disponibilidad, confiabilidad, performance, escalabilidad. Mapea elementos de las otras vistas a nodos de procesamiento.
- Vista de Casos de Uso o Escenarios: definición de procesos, agrupamiento en paquetes.

Como se puede observar este modelado descuida los temas referidos a Datos, Seguridad, Presentación, Despliegue y Sistema, esta última es muy importante para la presente investigación.

Microsoft propone:

- Vista Conceptual: Es usada para definir los requerimientos funcionales y la visión que los usuarios del negocio tienen de la aplicación y describir el modelo de negocio que la arquitectura debe cubrir. Esta vista muestra los subsistemas y módulos en los que se divide la aplicación y la funcionalidad que

brinda dentro de cada uno de ellos. Casos de Uso, Diagramas de Actividad, Procesos de Negocio Entidades del Negocio, entre otros.

- Vista Lógica: Muestra los componentes principales de diseño y sus relaciones de forma independiente de los detalles técnicos y de cómo la funcionalidad será implementada en la plataforma de ejecución. Los arquitectos crean modelos de diseño de la aplicación, los cuáles son vistas lógicas del modelo funcional y que describen la solución. Realización de los Casos de Uso, subsistemas, paquetes y clases de los casos de uso más significativos arquitectónicamente.
- Vista Física: Ilustra la distribución del procesamiento entre los distintos equipos que conforman la solución, incluyendo los servicios y procesos de base. Los elementos definidos en la vista física se "mapean" a componentes de software (servicios, procesos, entre otros.) o de hardware que definen más precisamente cómo se ejecutará la solución.
- Vista Implementación: Describe cómo se implementan los componentes físicos mostrados en vista de distribución agrupándolos en subsistemas organizados en capas y jerarquías, ilustra, además las dependencias entre éstos. Básicamente, se describe el mapeo desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (16).

Siemens Corporate Research propone:

- Vista conceptual: principales elementos de diseño y su interrelación.
- Vista de módulos: estructura funcional y de capas.
- Vista de ejecución: estructura dinámica.
- Vista de código: organización de código fuente, binarios y bibliotecas en el ambiente de desarrollo (17).

Este modelo también descuida la vista de sistema.

En el libro *"Software Systems Architecture"* (18) se propone:

- Vista funcional.
- Vista de información.
- Vista de concurrencia.
- Vista de desarrollo.
- Vista de despliegue.
- Vista operacional.

Este además de descuidar la integración y la seguridad, también obvia la vista de sistema.

Después de haber analizado cada uno de estos modelos se detectó que ninguno de ellos menciona, describe o explica la vista de sistema que es el principal objetivo de la presente investigación, por lo que se propone el

modelo de las 9+1 vistas desarrollada en la UCI, el cual propone un modelo arquitectónico donde se integran las corrientes más representativas y además se tiene en cuenta la vista a desarrollar.

Este modelo presenta nueve vistas y además una guía base de análisis arquitectónico (Figura 2), desde la perspectiva de procesos, que facilita el ordenamiento y priorización de estas vistas arquitectónicas, es además un elemento normativo, formativo y de auto evaluación del arquitecto (19).

Figura 1. Guía base de análisis arquitectónico.

Es por esto y porque en cada una de las vistas se caracteriza el escenario (condiciones específicas de funcionamiento, respuesta del sistema y características relevantes de su entorno) elementos que demuestran ampliamente la decisión tomada evitando ambigüedades.

1.7 Vista de Sistema de la Arquitectura

La Arquitectura de Sistema es una de las disciplinas más complejas dentro de la Arquitectura de Software, es responsable de definir correctamente cohesionados, acoplados e interrelacionados los elementos computacionales del producto, además de las principales interacciones, los conectores y las configuraciones a asumir por los elementos computacionales en función de los elementos del negocio que los mismos abstraen. La vista de sistema de la Arquitectura de Software, también representa una proyección simétrica de alto nivel, de los procesos de negocio o arquitectura de negocio que se trabaja, expresada en elementos, conectores, configuraciones y restricciones.

1.7.1 Elementos de la Vista de Sistema

La vista de Sistema está compuesta por un grupo de elementos que constituyen acciones a seguir para definir la misma. Inicialmente se debe caracterizar el producto atendiendo a: personal de desarrollo, dinamismo, cultura, tamaño del equipo y criticidad. Luego se priorizan los requisitos, y se estima su relevancia arquitectónica, para realizar un agrupamiento semántico de los requisitos priorizados. A continuación se hace un breve estudio de estos y otros elementos, con el fin de alcanzar mejoras en la toma de decisiones para desarrollar la propuesta de solución a la Vista de Sistema de la plataforma GESPRO.

1.7.1.1 Caracterización del producto

Para la caracterización de la Vista de Sistema existen un grupo de elementos importantes que influyen en la arquitectura de la Plataforma GESPRO 12.05 los cuales son:

- **Personal.** El por ciento del personal con capacidad, experiencia y posibilidades para enfrentar tareas es un aspecto de mucha importancia en un proyecto de producción de software, de esto depende mucho la posibilidad que tiene el grupo de desarrolladores de poder implantar o desarrollar un tipo de metodología u otra. Esta variable tiene su mayor importancia a la hora de adoptar métodos ágiles que requieren de un personal con niveles de experiencia medios-altos, capaces de comprender y adaptar los métodos y las técnicas empleadas.
- **Criticidad.** La criticidad del sistema resultada determinante cuando se habla de proyectos cuya falla puede traer pérdidas de vidas humanas, daño al medio ambiente y grandes pérdidas económicas, por lo que se hace necesario emplear métodos robustos capaces de mantener un rigor de requisitos y diseño adecuados para procesos de pruebas, verificación y validación.
- **Cultura.** El por ciento del personal con capacidad para enfrentarse a entornos caóticos resulta importante a la hora de adoptar métodos ágiles los cuales se basan fundamentalmente en el talento de las personas, un ambiente laboral con un control excesivamente normalizado y jerarquizado resultaría incómodo para llevar a cabo este tipo de práctica.
- **Tamaño.** La cantidad de personas de un proyecto resulta importante en dependencia de qué método ágil o robusto se vaya a poner en práctica. Para métodos ágiles donde es muy significativa una buena comunicación entre todos los integrantes del grupo de desarrollo esta resulta imposible con un número alto de personal en el proyecto.
- **Dinamismo.** El por ciento de cambios de requerimientos que pueden ocurrir en un mes tiene gran peso en los métodos ágiles preparados para enfrentar este tipo de situación, resultaría muy incómodo

enfrentar grandes cambios de requerimientos con métodos robustos que tienen grandes volúmenes de documentación y una verificación y validación de procesos continua.

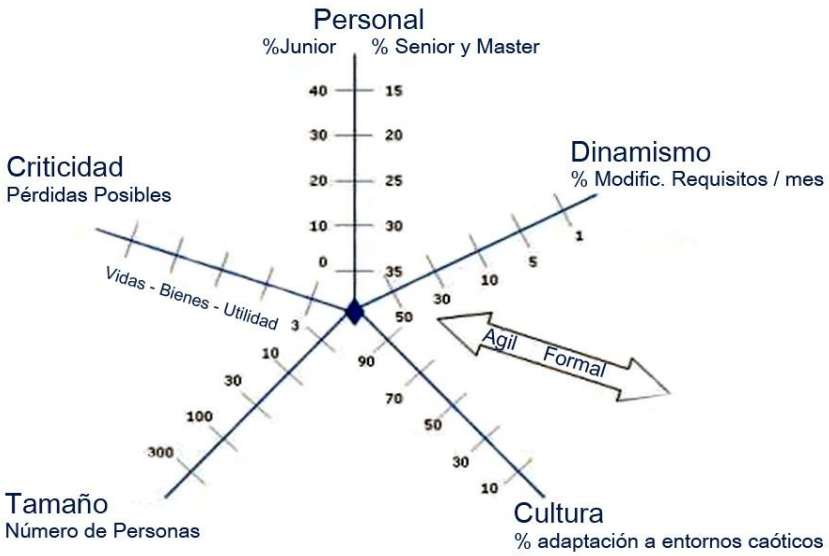

Figura 2. Gráfico para la selección de los modelos de desarrollo propuesto por Barry Boehm y Richard Turner.

Esta serie de elementos en dependencia de las características de la Plataforma GESPRO 12.05, se evalúan según la gráfica de la Figura 2.

1.7.1.2 Priorización de requisitos funcionales del sistema

En la actualidad es de vital importancia para el desarrollo de proyectos la priorización de los requisitos, para esto se relacionan los requerimientos funcionales que impactan la arquitectura del sistema, los que constituyen la esencia del negocio de la organización cliente. En GESPRO 12.05 se priorizan los requisitos según:

- El Grado de impacto en el negocio para el cliente (GI), es decir, que tan importante es el desarrollo de la funcionalidad para el cliente.
- El Grado de complejidad de la implementación (GC), es decir, que tan difícil es el desarrollo de la funcionalidad para el programador.
- El Grado de dependencia de otros requisitos (GD), si el requerimiento depende de otro u otros requisitos.
- El Grado de necesidad de persistencia de los conceptos que maneja el requisito (GCP), es decir, que tan importante es el requerimiento o lo que hace la funcionalidad.
- El Grado de impacto para la arquitectura (GIA), es decir, que nivel de impacto tiene para la arquitectura

del sistema.

Después de priorizarlos, teniendo en cuenta lo anterior se halla la relevancia arquitectónica de un requisito (RARF), que no es más que la multiplicación de la evaluación de los criterios de priorización por requisitos (RARF = GI * GC * GD * GCP * GIA).

1.7.1.3 Agrupamiento y Priorización de los requisitos priorizados

El agrupamiento es algo muy importante a la hora de ver cómo implementar cada uno de los requisitos priorizados del sistema, pues mediante esto se pueden desarrollar todos los requisitos que están relacionados y tienen un nivel de priorización mayor que los demás. En la Plataforma GESPRO 12.05, se agrupan los requisitos semánticamente. Se priorizan por su relevancia arquitectónica y se hace un listado de los requerimientos ordenados por su RARF de mayor a menor para su posterior implementación, luego se indica el grupo al que pertenece.

1.7.1.4 Identificación de los paquetes y componentes principales

Luego de haber listado de mayor a menor por su relevancia arquitectónica y de haber agrupado los requisitos en la Plataforma GESPRO 12.05, se lleva a cabo la identificación de paquetes y componentes, para esto se tienen en cuenta una serie de reglas que se indican a continuación:

- El nivel subsistema corresponde con la abstracción de los procesos globales del negocio enfocado a objetivos únicos en la organización. Los subsistemas son contenedores de módulos u otros subsistemas, que no son más que los paquetes o componentes que se identifiquen.
- A partir de los grupos construidos en la agrupación anterior se indican los subsistemas o paquetes principales de la solución. Cada uno de los grupos que se identificaron generan al menos un paquete.
- Cada uno de los subsistemas en dependencia de los requisitos que represente se clasificará en una de las cinco categorías siguientes (ver Figura 2):

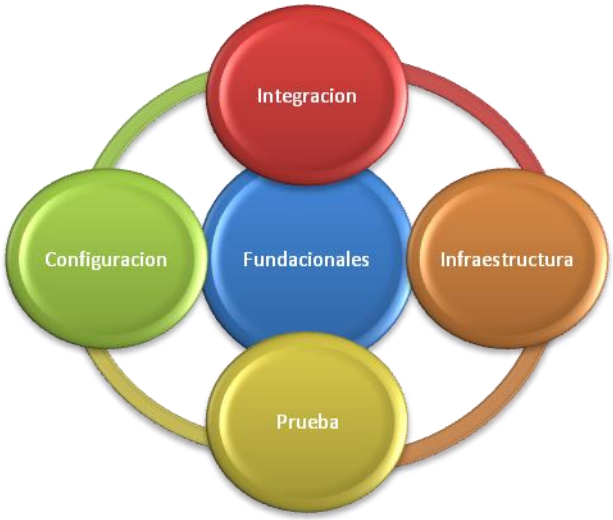

Figura 3. Categorías arquitectónicas que clasifican un elemento arquitectónico.

- ✓ Fundacional: 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. Es característico en ellos un alto impacto en el resto de las decisiones arquitectónicas de los otros grupos de componentes, ya que por contener los procesos núcleos y las principales agrupaciones conceptuales, a partir de sus abstracciones se definen las características de configuración y parametrización que deberá tener el sistema, así como los flujos y procesos de integración de los elementos arquitectónicos.
- ✓ Infraestructura: 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 abstraer las características no funcionales, con el objetivo de lograr una separación arquitectónica y una mejor especialización entre los diferentes grupos de arquitectos de soluciones, permitiéndose una mejor trazabilidad en la arquitectura del sistema de la arquitectura del negocio. Generalmente son constituidos en frameworks tecnológicos que trabajan como especie de mano mágica en la solución arquitectónica y permiten una alta reutilización de las características tecnológicas, además de una alta productividad en el desarrollo, es el grupo de componentes más asociado con la arquitectura tecnológica. Ejemplos de componentes en estas categorías son: componentes para el tratamiento de las trazas, las excepciones, la concurrencia de información, el caché de datos, la gestión de acceso, la validación de contrato, entre otros.
- ✓ 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.

- ✓ 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 con otras plataformas tecnológicas afines al sistema en cuestión o de otro tipo, como las plataformas ofimáticas, pasarelas bancarias, sistema de gestión de entidades o sistemas legados de la organización que requieren interacción con el sistema en modelación. Esta categoría asume además los componentes asociados a características horizontales en la arquitectura de sistema, a la implementación de patrones de integración para mejorar la calidad del diseño arquitectónico con el objetivo de eliminar lazos de dependencia funcional entre componentes o para implementar políticas de reutilización especificas de la arquitectura de sistema. Esta característica posee un alto impacto en los elementos tecnológicos del diseño, pues en ellos recae solo la responsabilidad de la actividad del proceso, pero el elemento tecnológico debe garantizarlo el grupo de componentes arquitectónicos tecnológicos.

- ✓ Pruebas: 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, para a partir de una aceptación de la maqueta del producto subcontratar el resto de las características del sistema, los niveles de parametrización, interoperabilidad e integración a incluir en la solución.

- En cada subsistema se identificarán además los componentes de nivel 1 o componente COTS (*Commercial Off The Shelf*) y de nivel 2 que lo forman, los componentes que generalmente están relacionados con actividades que son repetidas con alta frecuencia en la mayoría de las áreas de proceso que se modelan y que una vez proyectados en el diseño arquitectónico, resultan funcionalidades de alta incidencia en la colaboración y de elevada frecuencia de solicitud en la ejecución del sistema.

- La evaluación de los componentes de 2do nivel y los componentes COTS se realizará siguiendo las fórmulas o dominio de la variable de los siguientes criterios:

  - ✓ Grado de impacto en el negocio para el cliente a partir del cubrimiento de los requisitos del sistema donde ¨*n*¨ cantidad de requisitos asociados al componente. (GIN) y se calcula GIN = $\frac{\sum_1^n RARFi}{n}$.

  - ✓ Grado de dependencias de otros componentes asociados a él (GDC) con un dominio de la variable de 1 a 7.

- ✓ Grado de complejidad en la implementación (GC) con un dominio de la variable de 1 a 5.
- ✓ Grado de necesidad de persistencia de los conceptos que maneja el componente (ejemplo cantidad de tablas en la BD que afecta) (GCP) con un dominio de la variable de 1 a 5.
- ✓ Grado de entidades o conceptos de negocio que afecta (GEA) con un dominio de la variable de 1 a 5.
- ✓ Grado de dependencia de clases controladoras (GDAC) con un dominio de la variable de 1 a 5.
- ✓ El Índice de reutilización (IR) que se calcula IR = (A + B + C + D) / 4. Dónde los valores que se le debe dar a las variables A, B, C y D están en la siguiente tabla.

Tabla 1. Umbral cuantitativo

A - Extensión	
10	Se reutiliza completamente el componente núcleo.
8	Basta con implementar el patrón *Adapter*, no existe necesidad de modificar las interfaces del componente base, pues el formato coincide.
6	Implica transformaciones de formato.
4	Implica modificar el código fuente para poder ser extendido.
2	Implica modificar el código, no se conoce cómo está hecho, requiere estudiar el componente, para poder ser entendido.
0	No reutilizable.
B- Mantenimiento	
10	No existe necesidad de dar mantenimiento al componente.
8	El mantenimiento no implica modificaciones importantes, puede resolverse en 2 horas hombre de trabajo.
6	Implica modificaciones de interfaz y algunas lógicas internas.
4	Implica grandes modificaciones, solo se reutilizan algunas funcionalidades, su importancia recae en la complejidad y esfuerzo de implementación de las mismas.
2	Implica grandes modificaciones, sólo se reutilizan algunas funcionalidades, no se conoce el componente ni se dominan los conocimientos suficientes, necesario entrenar al equipo, su importancia recae en la complejidad y esfuerzo de implementación de las mismas.
0	No reutilizable.
C -Parametrización	
10	No requiere parametrización.
8	No requiere parametrización.

6	No requiere parametrización.
4	Requiere parametrización baja.
2	Requiere parametrización compleja.
0	No reutilizable.
D- Adquisición	
10	Registrado en el repositorio de componentes.
8	Se puede adquirir de manera gratuita desde alguna comunidad.
6	Requiere adquisición, con documentación.
4	Requiere adquisición, con documentación deficiente.
2	Requiere adquisición, sin documentación.
0	No se puede adquirir.

Todos estos criterios dan al traste con la significancia arquitectónica de un componente (SAC) que se calcula con la multiplicación de todos los criterios SAC = GIN * GDC * GC * GCP * GEA * GDFC * GD * AC * IR.

Después de realizar cada una de estas reglas quedan conformados los subsistemas y sus componentes o paquetes que a su vez son subsistemas también.

1.7.1.5 Representación de la dependencia entre paquetes

Otra de las acciones a seguir es el diseño del sistema descomponiendo sus partes hasta el nivel necesario identificando claramente los subsistemas que se encuentran relacionados en serie y en paralelo y las relaciones entre ellos. Para esto se realiza un diagrama (ver Figura 4 y Figura 5) donde se ve la dependencia entre paquetes, conociéndose que cada uno de los paquetes son subsistemas.

Figura 4. Vista de dos subsistemas conectados en serie.

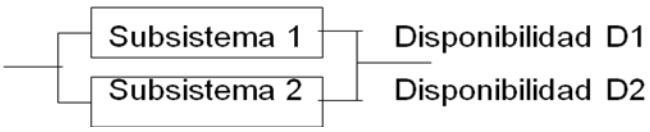

Figura 5. Vista de dos subsistemas conectados en paralelo.

1.7.1.6 Determinación de los escenarios de la vista

La Vista de Sistema propone 12 escenarios para un mejor desarrollo de la aplicación, estos son:

- Conceptos globales
- Nomencladores
- Nomenclador de nomencladores para la multientidad
- Buscador de conceptos en un dominio
- Proveedor de perfiles de personas
- Gestión de entidades, estructura organizacional
- Configuración de flujos de trabajo
- Operaciones ajustadas en fechas
- Compilador de expresiones matemáticas
- Operaciones con multimoneda
- Configuración de reglas contables
- Tratamiento de excepciones

Estos escenarios pueden ser aplicables o no a la plataforma GESPRO 12.05, cada uno de estos escenarios se analizan y se evalúan teniendo en cuenta las descripciones de los mismos y las características de la plataforma para saber si se adecuan a la propuesta de solución, además presentan un patrón de solución que puede contribuir a un mejor desarrollo de la aplicación y también se pueden observar las restricciones que tiene este último.

1.8 Conclusiones del Capítulo 1

El estudio realizado sobre el estado del arte, permitió profundizar en los principales conceptos de Arquitectura de Software, así como los aspectos más trascendentes asociados a esta como estilos, patrones arquitectónicos entre otros. Se estudiaron las principales tendencias que a nivel mundial tratan la Arquitectura de Software, con énfasis en las seis corrientes más reconocidas, se analizaron los modelos arquitectónicos más utilizados y se evidenció que el de las 9 + 1 vistas es el que más se ajusta a la propuesta de solución. Se abordaron también los aspectos más relevantes que conforman la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05, sobre la cual se presenta la propuesta de solución de la presente investigación.

Capítulo 2. Propuesta de Solución

2.1 Introducción

En el presente capítulo se presenta la propuesta de solución al problema planteado para la investigación. Se gestionan los requisitos funcionales del sistema, a través de la priorización de estos, de acuerdo a su relevancia arquitectónica para luego ser agrupados en clases o paquetes y se crean los subsistemas presentes en la Plataforma GESPRO 12.05. También se hace una selección de los escenarios que están acordes con la Vista de Sistema de la Plataforma GESPRO 12.05.

2.2 Acciones de la Vista Sistema:

- Caracterización del producto según la vista.
- Priorización de los requisitos funcionales del sistema.
- Agrupamiento y priorización de los requisitos priorizados.
- Identificación de los paquetes y componentes principales.
- Representación de la dependencia entre paquetes.
- Descripción de los escenarios y patrones de solución.
- Realizar autoevaluación de la vista.

2.2.1 Caracterización del producto

Después de evaluar la aplicación atendiendo al gráfico (Figura 1) y a entrevistas realizadas a expertos del proyecto, se seleccionó para cada uno de los cinco elementos los valores más apropiados. Con el análisis realizado se elaboró la siguiente tabla donde se muestran cada uno de los elementos básicos de la Plataforma GESPRO 12.05 que influyen en la arquitectura:

Tabla 2. Elementos básicos de la Plataforma GESPRO 12.05 que influyen en la arquitectura

Nombre de la variable	Valor
Nivel de especialización del personal	30 – *Senior,* 30 – *Junior*
Dinamismo de los requisitos	50%
Cultura de la organización	50%
Tamaño del equipo de desarrollo previsto	30
Criticidad de la aplicación	Utilidad

Como se puede observar en la Tabla 2, el nivel de especialización del personal *Junior* o principiantes es de 30 y el de *Senior* o *Masters* es de 30 también, este elemento tiene un valor máximo de hasta 35 por ciento de especialización por lo que se puede seguir mejorando. Se puede observar como los requisitos son muy dinámicos, ya que se le da un valor de 50, lo que quiere decir que se modifican con frecuencia cada mes. Se aprecia también como la organización es en gran medida adaptable a los cambios a entornos caóticos, se puede observar también el tamaño del equipo de desarrollo, dentro del cual se encuentran varios másteres en ciencias, doctores en ciencias e ingenieros y se tiene en cuenta además que puede haber pérdidas en utilidades.

2.2.2 Priorización de requisitos funcionales del sistema.

Para esta acción se entrevistaron a especialistas de la plataforma, obteniéndose como resultado que de la totalidad de los requisitos se identificaron 179 que forman parte de la columna vertebral del sistema. Después de esto se priorizaron los mismos midiendo los criterios expuestos en el sub-epígrafe 1.7.1.2 y se le halló la relevancia arquitectónica a cada uno de ellos. A continuación se muestran ejemplos de cada uno de los módulos a los que se le identificaron requisitos priorizados:
Ejemplo para el módulo de Alcance y Tiempo:
Nombre del requisito 5: RF Creación de procesos del centro.

- Identificador del proceso: Procesos para el Alcance y Tiempo.
- RARF: 735.
- Requisitos predecesores del requisito en análisis: No tiene.

Ejemplo para el módulo de Control y Seguimiento:
Nombre del requisito 89: RF Consultar planificación del proyecto.

- Identificador del proceso: Procesos para el control y seguimiento.
- RARF: 1225.
- Requisitos predecesores del requisito en análisis: No tiene.

Ejemplo para el módulo de Gestión Logística:

Nombre del requisito 151: RF Listar Recursos a nivel de centro.

- Identificador del proceso: Procesos para la Gestión Logística.
- RARF: 2450.
- Requisitos predecesores del requisito en análisis:
    - ✓ RF Crear Recursos a nivel de centro.

Ejemplo para el módulo de Trabajo Colaborativo:

Nombre del requisito 159: RF Visualizar histórico de la Wiki.

- Identificador del proceso: Procesos para Trabajo Colaborativo.
- RARF: 6125.
- Requisitos predecesores del requisito en análisis:
    - ✓ RF Crear página Wiki.

Ejemplo para el módulo de Gestión Documental:

Nombre del requisito 164: RF Listar ficheros.

- Identificador del proceso: Procesos para la Gestión Documental.
- RARF: 4900.
- Requisitos predecesores del requisito en análisis:
    - ✓ RF Crear Fichero.

Ejemplo para el módulo de Recursos Humanos:

Nombre del requisito 167: RF Guardar los datos especificados por el usuario.

- Identificador del proceso: Procesos para los Recursos Humanos.
- RARF: 4900.
- Requisitos predecesores del requisito en análisis:
    - ✓ RF Crear miembros de la organización, registrarse en la aplicación.

Ejemplo para el módulo de Administración:

Nombre del requisito 177: RF Modificar las fases del proyecto.

- Identificador del proceso: Procesos para la Administración.
- RARF: 6125.
- Requisitos predecesores del requisito en análisis:

- ✓ RF Crear un nuevo proyecto.
- ✓ RF Crear las fases del proyecto

Después de haberle hallado el RARF, los requisitos predecesores y el proceso al que pertenece, a cada uno de los 179 requerimientos, se da paso a la acción de agrupamiento y priorización de requisitos priorizados.

2.2.3 Agrupamiento y priorización de los requisitos

Para la ejecución de este paso se tuvo en cuanta la serie de reglas que se muestran en el sub-epígrafe 1.7.1.3. Teniendo en cuenta que se priorizaron requisitos de 7 módulos, se agruparon los mismos en 9 grupos, debido a que los requisitos de los módulos de Alcance y Tiempo y los de Control y Seguimiento fueron separados, en Alcance, Tiempo, Control y en Seguimiento, además de esto cada uno de los 9 grupos tiene una baja dependencia de los otros. También se ordenaron de mayor a menor los requerimientos funcionales según su RARF, teniendo así una priorización de los requisitos priorizados.

A continuación se muestra un ejemplo de la tabla con los grupos semánticos y los requisitos priorizados ordenados por su RARF.

Tabla 3. Agrupamiento semántico de los requisitos.

Listado de requisitos priorizados por la arquitectura (Identificador)	RARF (ordenado de forma descendente)	Grupo al que pertenece (nombre o identificador)
161	6125	Trabajo Colaborativo
146	6125	Seguimiento
170	6125	Recursos Humanos
175	6125	Administración
165	4900	Gestión Documental
47	4500	Tiempo
36	3000	Alcance
103	2700	Control
151	2450	Gestión Logística

2.2.4 Identificación de los paquetes y componentes principales

Después de analizar las reglas expuestas en el sub-epígrafe 1.7.1.4, se identificaron 9 paquetes o componentes, estableciéndose una correspondencia entre componentes y grupos (para cada grupo un

componente), anteriormente formados y los cuales se nombran: Alcance, Tiempo, Control, Seguimiento, Logística, Colaborativo, Documental, Recursos Humanos y Administración. Cada uno de los componentes o paquetes definidos son tratados como subsistemas y están compuestos por los requisitos priorizados de los módulos de la plataforma GESPRO 12.05.

A continuación se muestra una tabla con los componentes o subsistemas creados y la cantidad de requisitos que tiene cada uno:

Tabla 4. Componentes

Nombre o Identificador del Componente	Cantidad de Requisitos
Alcance	62
Tiempo	27
Control	33
Seguimiento	28
Logística	7
Colaborativo	6
Documental	2
Recursos Humanos	8
Administración	6

Finalmente a cada uno de los subsistemas se le clasificó según su categoría, se indicó a los grupos de requisitos que representa, se identificó que tipo de componente es y se le halló su significancia arquitectónica como se muestra a continuación:

Subsistema: Alcance.

- Grupo de requisitos que representa: Procesos de Alcance y Tiempo.
- Categoría del subsistema o paquete:

✓	Fundacional	□	Integración
□	Configuración	□	Prueba
□	Infraestructura		

- Clasificación del componente:

□	Componente COTS
✓	Componente de negocio no COTS
□	Componente conector

- Significancia Arquitectónica del Componente:

Tabla 5. Alcance

Criterios	Dominio de la variable	Evaluación
GIN		1007
GDC	1..7	3
GC	1..5	4
GCP	1..5	4
GEA	1..5	3
GDAC	1..5	1
GDFC	1..5	3
IR		8.5
SAC		3697704

Subsistema: Tiempo.

- Grupo de requisitos que representa: Procesos de Alcance y Tiempo.
- Categoría del subsistema o paquete:

✓	Fundacional	□	Integración
□	Configuración	□	Prueba
□	Infraestructura		

- Seleccione la clasificación del componente:

□	Componente COTS
✓	Componente de negocio no COTS
□	Componente conector

- Significancia Arquitectónica del Componente:

Tabla 6. Tiempo

Criterios	Dominio de la variable	Evaluación
GIN		2900
GDC	1..7	3
GC	1..5	4
GCP	1..5	4
GEA	1..5	3
GDAC	1..5	1

GDFC	1..5	3
IR		8.5
SAC		10648800

Subsistema: Control.

- Grupo de requisitos que representa: Procesos de Control y Seguimiento.
- Categoría del subsistema o paquete:

✓	Fundacional	□	Integración
□	Configuración	□	Prueba
□	Infraestructura		

- Clasificación del componente:

□	Componente COTS
✓	Componente de negocio no COTS
□	Componente conector

- Significancia Arquitectónica del Componente:

Tabla 7. Control

Criterios	Dominio de la variable	Evaluación
GIN		1259
GDC	1..7	3
GC	1..5	5
GCP	1..5	4
GEA	1..5	3
GDAC	1..5	1
GDFC	1..5	3
IR		8
SAC		5438880

Subsistema: Seguimiento.

- Grupo de requisitos que representa: Procesos de Control y Seguimiento.
- Categoría del subsistema o paquete:

✓ Fundacional □ Integración

□ Configuración □ Prueba

□ Infraestructura

- Clasificación del componente:

□ Componente COTS

✓ Componente de negocio no COTS

□ Componente conector

- Significancia Arquitectónica del Componente:

Tabla 8. Seguimiento

Criterios	Dominio de la variable	Evaluación
GIN		3570
GDC	1..7	3
GC	1..5	5
GCP	1..5	4
GEA	1..5	3
GDAC	1..5	1
GDFC	1..5	3
IR		8
SAC		15422400

Subsistema: Logística.

- Grupo de requisitos que representa: Procesos de Gestión Logística.
- Categoría del subsistema o paquete:

✓ Fundacional □ Integración

□ Configuración □ Prueba

□ Infraestructura

- Clasificación del componente:

□ Componente COTS

✓ Componente de negocio no COTS

□ Componente conector

- Significancia Arquitectónica del Componente:

Tabla 9. Logística

Criterios	Dominio de la variable	Evaluación
GIN		1925
GDC	1..7	3
GC	1..5	4
GCP	1..5	4
GEA	1..5	3
GDAC	1..5	2
GDFC	1..5	2
IR		7.8
SAC		8648640

Subsistema: Colaborativo.

- Grupo de requisitos que representa: Procesos de Trabajo Colaborativo.
- Categoría del subsistema o paquete:

✓	Fundacional	□	Integración
□	Configuración	□	Prueba
□	Infraestructura		

- Clasificación del componente:

□	Componente COTS
✓	Componente de negocio no COTS
□	Componente conector

- Significancia Arquitectónica del Componente:

Tabla 10. Colaborativo

Criterios	Dominio de la variable	Evaluación
GIN		3920
GDC	1..7	4
GC	1..5	5
GCP	1..5	5
GEA	1..5	4
GDAC	1..5	3
GDFC	1..5	2
IR		8.5

SAC	79968000

Subsistema: Documental.

- Grupos de requisitos que representa: Procesos de Gestión Documental.
- Categoría del subsistema o paquete:

✓	Fundacional	□	Integración
□	Configuración	□	Prueba
□	Infraestructura		

- Clasificación del componente:

□	Componente COTS
✓	Componente de negocio no COTS
□	Componente conector

- Significancia Arquitectónica del Componente:

Tabla 11. Documental

Criterios	Dominio de la variable	Evaluación
GIN		4900
GDC	1..7	6
GC	1..5	3
GCP	1..5	4
GEA	1..5	5
GDAC	1..5	4
GDFC	1..5	2
IR		9
SAC		127008000

Subsistema: Recursos Humanos.

- Grupo de requisitos que representa: Procesos de Recursos Humanos.
- Categoría del subsistema o paquete:

□	Fundacional	□	Integración
✓	Configuración	□	Prueba
□	Infraestructura		

- Clasificación del componente:

□ Componente COTS

✓ Componente de negocio no COTS

□ Componente conector

- Significancia Arquitectónica del Componente:

Tabla 12. Recursos Humanos

Criterios	Dominio de la variable	Evaluación
GIN		4900
GDC	1..7	6
GC	1..5	3
GCP	1..5	4
GEA	1..5	5
GDAC	1..5	4
GDFC	1..5	4
IR		8.2
SAC		231436800

Subsistema: Administración.

- Grupos de requisitos que representa: Procesos de Administración.
- Categoría del subsistema o paquete:

□ Fundacional

✓ Configuración

□ Infraestructura

□ Integración

□ Prueba

- Clasificación del componente:

□ Componente COTS

✓ Componente de negocio no COTS

□ Componente conector

- Significancia Arquitectónica del Componente:

Tabla 13. Administración

Criterios	Dominio de la variable	Evaluación
GIN		4696
GDC	1..7	2
GC	1..5	5

GCP	1..5	5
GEA	1..5	4
GDAC	1..5	2
GDFC	1..5	5
IR		9
SAC		84528000

2.2.5 Representación de la dependencia entre paquetes

En este epígrafe se presenta un diagrama de la relación existente entre los componentes o subsistemas, y de cómo están relacionados, de forma paralela, los subsistemas de Control y Seguimiento, y los de Alcance y Tiempo; además se muestra la relación en serie de los subsistemas Documental, Colaborativo, Recursos Humanos (R.R.H.H.) y Logística; inicia el diagrama el de Administración conectado en serie con el de Control y el de Seguimiento.

Figura 6. Diagrama de componentes y subsistemas

Como se puede observar en la Figura 6 el nivel de disponibilidad comienza desde el subsistema de Administración que da inicio a la relación entre componentes o subsistemas.

2.2.6 Determinación de los escenarios de la vista

En este epígrafe se seleccionan los escenarios y patrones de solución requeridos para el sistema GESPRO, y se describen los específicos para la solución, que no han sido descritos en el documento base.

Escenario: Conceptos globales.

La arquitectura tecnológica de la plataforma proveerá algún mecanismo para la configuración dinámica y centralizada, variables globales que son aspectos muy importantes en los frameworks ya que agrupan información común conceptualizada que se maneja en todo el sistema, o sea en todos los módulos y sesiones de una aplicación y la Plataforma GESPRO 12.05 utiliza frameworks para su desarrollo, por tanto este escenario es aplicable a la solución propuesta.

Patrón de Solución del escenario:

- Por lo general todo framework que se implementa contiene un componente de este tipo para que cada parte del sistema pueda acceder de forma fácil y sencilla a datos que oferte otra sin incurrir en códigos engorrosos que implican gastos de tiempo y esfuerzos para los programadores.
- La implementación de estos contenedores de información común permiten que los datos sean almacenados una sola vez y actualizados en el momento conveniente, así cuando un componente necesite de esta información la toma del contenedor sin tener que estar realizando peticiones constantes a los servicios que oferten los demás componentes.

Restricciones del patrón solución:

En caso de no manejarse adecuadamente los conceptos globales se puede provocar que el sistema tenga un elevado costo de rendimiento.

Por otra parte no se debe abusar de los conceptos globales porque afectan el encapsulamiento de la información.

Atributos de calidad que impacta:

Eficiencia, Funcionalidad, Mantenibilidad.

Escenario: Nomencladores.

La gestión de datos en los sistemas de información empresarial suele estar determinada por distintas clasificaciones se hace necesaria la introducción de nomencladores. Con frecuencia los nomencladores son utilizados en diferentes subsistemas de un gran sistema de gestión de información.

La configuración de los nomencladores de un sistema de gestión puede implicar la relación de estos con otros conceptos del sistema, sean nomencladores o no. La Plataforma GESPRO utiliza diferentes nomencladores por lo que este escenario es aplicable a la solución propuesta.

Patrón de solución del escenario:

- Se centralizó la información común en único espacio, y se brindó mecanismos de estructuración, consulta y gestión de los nomencladores. Para ello se propuso implementar un componente especializado que brinde información a los diferentes subsistemas.
- Se cestionó las relaciones dinámicamente para el uso de nomencladores en diversas funcionalidades elementos que pueden ser manejados desde el diseño de la base de datos.

Restricciones del patrón solución:

Implica que se maneje adecuadamente la concurrencia en el acceso al componente nomencladores protegiendo incluso la confiabilidad de la información.

Atributos de calidad que impacta:

Funcionalidad, Mantenibilidad

Escenario: Configuración de flujos de trabajo.

Existen conceptos dentro de los sistemas de gestión cuyo proceso depende de las transiciones de estados, básicamente flujos de trabajo aplicados a objetos concretos del negocio. Este escenario es frecuente en sistemas de gestión documental entre otros. La Plataforma GESPRO 10.05 trabaja con documentos de diferentes funciones por lo que este escenario es aplicable a la solución propuesta.

Patrón de Solución del escenario:

- Se permite configurar arquitecturas de flujos de trabajo expresando para ello tres elementos arquitectónicos concretos (actividades).
- Las actividades, se clasificaron en actividades simples, compuestas, actividades provistas (las que vienen como resultado de un evento, una traza o un servicio web) y condicionales.
- Los otros elementos que se configuraron en el flujo de trabajo fueron las reglas de control, es decir reglas pre-condicionales de cada actividad del flujo, para ello se utilizó el mismo mecanismo definido en el marco de solución arquitectónica para validación, o se estableció uno para el marco de flujos de trabajo.
- El otro concepto que se configuró del flujo de trabajo fue el flujo en sí, el cual queda compuesto por la secuencia de actividades y reglas de control de cada actividad, en una configuración arquitectónica de flujo de trabajo que podrá ser exportado, probado o simulado mediante el marco de gestión de flujos de trabajo, de manera que tecnológicamente se cuente con una capacidad tecnológica de programar programas de flujos de trabajo, estos pueden ser persistidos, exportados e instanciado en cualquier subrutina secuencial no interna de un flujo de trabajo, permitiendo tecnológicamente y de manera transparente para el programador la instanciación del marco que se describe.
- Es importante mencionar que se definió además que el marco para la gestión de flujos de trabajo, debe establecer algún mecanismo para persistir el estado de un flujo de trabajo, implementar algún mecanismo de patrón memento para guardar el historial de las actividades y de esta forma permitir que el flujo de trabajo pueda ser además recorrido hacia delante y hacia detrás.
- Los flujos de trabajos además pueden ser configurados su carácter transaccional, integrando este marco al marco de transacciones, excepciones e inversión de control según sea el caso de aplicación que finalmente se decida.

Restricciones del patrón solución:

Esta solución debe evitar la creación de cuellos de botella en la solución, además debe ser programada para

un elevado nivel de concurrencia y teniendo en cuenta que no se trata de garantizar un acceso único al objeto o recurso sino de proveer información rápida de metadatos de un objeto para su futura gestión.

Atributos de calidad que impacta:

Mantenibilidad, Funcionalidad.

Escenario: Operaciones con multimoneda.

Los sistemas de gestión deben permitir la ejecución de transacciones financieras en cualquier moneda, registrando siempre la conversión para la moneda rectora de la aplicación. Cuba es un país donde existe la doble moneda por lo que se dificulta la contabilidad. Las transacciones financieras de la Plataforma GESPRO 12.05 se realizan en USD y en CUC por lo que este escenario es aplicable a la solución propuesta.

Patrón de Solución del escenario:

Se permite registrar operaciones en cualquiera de las dos monedas, y se permiten las posteriores reevaluaciones de las cuentas en correspondencia con la diferencia de cambio de tasas.

Atributos de calidad que impacta:

Funcionalidad, Mantenibilidad.

2.3 Conclusiones del Capítulo 2

Con la realización de este capítulo se logró desarrollar la propuesta de solución a la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05. Para esto se realizó una caracterización del producto midiendo los elementos básicos que influyen en la arquitectura de un Sistema. Se identificaron los requisitos funcionales priorizados y se le calculó su relevancia arquitectónica, para luego agruparlos semánticamente y priorizarlos por su RARF. Después se crearon diferentes Paquetes o Subsistemas y se realizó un gráfico con la relación entre los mismos y finalmente se identificaron los escenarios aplicables a la solución propuesta de la presente investigación.

Capítulo 3. Validación de la propuesta

3.1 Introducción

En el presente capítulo se realizó un estudio de los diferentes métodos de validación para ver cuál se adecua a la propuesta de solución. También se hizo una encuesta a especialistas del proyecto para observar los resultados una vez aplicada la solución que propone la presente investigación, finalmente se realizó un análisis de la misma para valorar si es factible su aplicación.

3.2 Métodos de validación

Se analizaron varios tipos de evaluaciones que existen, para aplicarlos a la propuesta de solución. Entre ellas están:

Grupo focal:

- Básicamente es la selección de un grupo de personas con conocimientos sobre el tema que se debatirá, deberán ser especialistas, expertos, de distintos niveles y categorías, que se reúnen en un lugar, para discutir sobre el procedimiento, siendo este debate dirigido por los autores, y centrado en lo que se quiere conocer sobre el procedimiento.

- Imposibilidad de realizar esta actividad por falta de personal.

Método de consulta a expertos. Método Delphi:

- Este desde sus inicios en los años 50 ha sido utilizado frecuentemente como sistema para obtener información sobre las ocurrencias de un fenómeno en el futuro. Consiste en la selección de un grupo de expertos a los que se les encuesta su opinión sobre cuestiones referidas a sucesos del futuro. El método se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos, obtenido, encuestando a este grupo mediante un cuestionario. Es un método fiable y muy utilizado actualmente pero necesita del estudio de competencias en los participantes del panel.

- No existe posibilidad de aplicarlo a la propuesta ya que, el personal consta con poca experiencia referente al tema.

Pre-experimento:

- Los llamados pre-experimentos son aquellos en los cuales no existe un grupo de control (patrón o testigo) para comparar. Por tanto, no hay, o se reducen las posibilidades de manipular las variables independientes y las conclusiones son extraídas en el mejor de los casos por la variación de la variable dependiente en relación con su historia anterior.

- Este fue el método seleccionado para la validación por las ventajas y oportunidades que ofrece. De esta manera se refleja la variación que ha sufrido la variable dependiente desde la aplicación de la propuesta.

3.2 Elaboración de la encuesta

La siguiente encuesta fue realizada a miembros del grupo de trabajo de la plataforma GESPRO, con el objetivo de medir la variación de la variable dependiente, para la validación de la solución propuesta. Los miembros que respondieron las preguntas tenían un previo conocimiento del contenido de la misma.

Encuesta para conocer el estado de la Vista Sistema antes de la aplicación de la propuesta:
1- ¿El levantamiento de los requisitos se hacía de forma formal?
_0 _1 _2 _3 _4 _5
2- ¿Los requisitos eran consecuentes con los requisitos reales que se desarrollaban en el proyecto?
_0 _1 _2 _3 _4 _5
3- ¿La actualización de los documentos era acorde con los requisitos?
_0 _1 _2 _3 _4 _5
4- ¿Se hacia una priorización de los requisitos para su posterior desarrollo?
_0 _1 _2 _3 _4 _5
5- ¿Se medía la relevancia arquitectónica de los requisitos?
_0 _1 _2 _3 _4 _5
6- ¿Se agrupaban semánticamente para una mayor calidad de desarrollo?
_0 _1 _2 _3 _4 _5

Encuesta para el estado de la Vista Sistema después de la aplicación de la propuesta:
1- ¿El levantamiento de los requisitos se hace de forma formal?
_0 _1 _2 _3 _4 _5
2- ¿Los requisitos son consecuentes con los requisitos reales que se desarrollaban en el proyecto?
_0 _1 _2 _3 _4 _5
3- ¿La actualización de los documentos es acorde con los requisitos?
_0 _1 _2 _3 _4 _5
4- ¿Se hace una priorización de los requisitos para su posterior desarrollo?

_0 _1 _2 _3 _4 _5

5- ¿Se mide la relevancia arquitectónica de los requisitos?

_0 _1 _2 _3 _4 _5

6- ¿Se agrupan semánticamente para una mayor calidad de desarrollo?

_0 _1 _2 _3 _4 _5

Haciéndose un análisis de la encuesta aplicada al personal de la Plataforma GESPRO y sus respuestas, se puede apreciar la variación de la variable dependiente de la propuesta de solución, a través de la siguiente tabla donde se muestra el por ciento de realización de acuerdo a los criterios empleados en cada una de las preguntas de la encuesta.

Tabla 14. Comparación en por ciento del antes y después de la aplicación de la propuesta de solución.

Criterio	Antes (%)	Ahora (%)
Levantamiento formal de los requisitos	40	80
Consecuencia con los requisitos reales	52	93
Actualización de los documentos	40	100
Priorización de los requisitos	40	100
Relevancia arquitectónica	47	93
Agrupamiento semántico	47	100

En el siguiente gráfico se muestra la variación sufrida por los diferentes criterios (%), antes y después de la aplicación de la propuesta de solución al problema de la investigación.

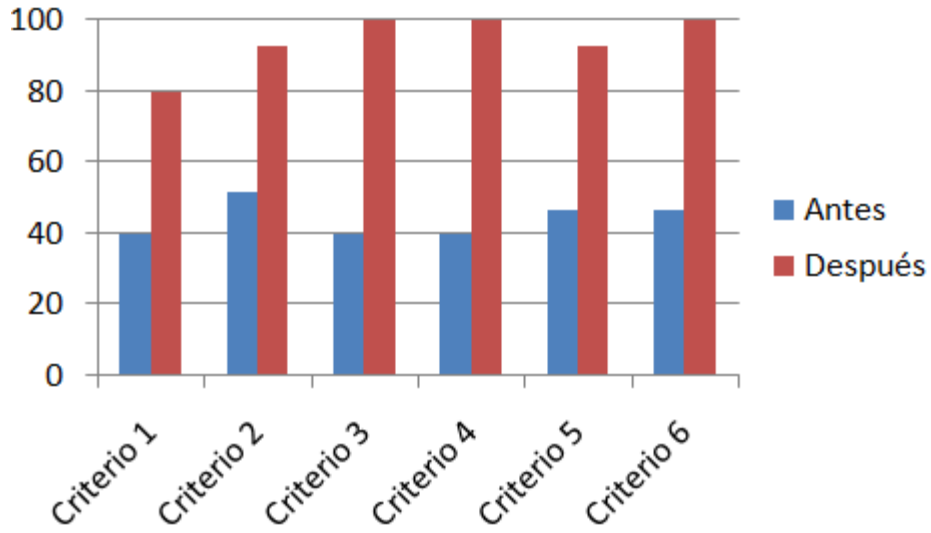

Figura 7. Gráfico de resultados después de la aplicación de la encuesta.

Analizando la tabla y el gráfico referente a ella, se puede apreciar que hubo un completo mejoramiento de los criterios evaluados después de la aplicación de la propuesta de solución con respecto a la situación anterior de la plataforma, aunque también se puede observar que en algunos de los criterios no se alcanzó un cien por ciento de mejoras.

3.3 Conclusiones del Capítulo 3

Con la realización de este capítulo culminó la propuesta de solución de la Vista de Sistema de la Arquitectura de la Plataforma GESPRO 12.05. Se validó la propuesta, demostrándose la mejora de la planificación de la calidad de la documentación digital y la gestión de los requisitos.

Conclusiones Generales

La investigación realizada en la elaboración de este trabajo de diploma y el análisis de la documentación estudiada permitió el cumplimeinto de los objetivos trazados. Después del estudio realizado y las razones expuestas en el contenido, se consiguió establecer un marco teórico para la Arquitectura de Software en la actualidad.

Se desarrolló la propuesta de solución para la Vista de Sistema de la Arquitectura de la Plataforma GESPRO, se logró caracterizar la solución vista desde los elementos básicos que influyen en la arquitectura que la forman. Se definieron además los paquetes y subsistemas para un posterior desarrollo de los requisitos priorizados del mismo. Se realizó un análisis de los escenarios que están presentes en la Vista de Sistema de la Arquitectura. Finalmente se pudo desarrollar la Vista de Sistema de la Arquitectura de GESPRO 12.05, cumpliéndose de esta manera el objetivo general de la investigación.

Después de la implantación fue posible demostrar la funcionalidad de la misma pues se logró mejorar la calidad de la documentación de los requisitos funcionales.

Recomendaciones

Recomendaciones

Se recomienda:

- El refinamiento constante de la arquitectura propuesta, durante el ciclo de desarrollo.
- Continuar la investigación sobre el resultado obtenido y proponerlo como base referencial o material auxiliar en el desarrollo de futuros trabajos asociados al objeto de estudio de Arquitectura de Software.
- Aplicarle la propuesta de solución a los nuevos requisitos funcionales del sistema que no han sido incluidos en la investigación.

Glosario de Términos

ADLs: *Architecture Description Language* (Lenguajes de Descripción Arquitectónica). Lenguaje descriptivo de modelado que se focaliza en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos.

CMM: Modelo de evaluación de los procesos de una organización. La visión general de los modelos basados en la madurez de las capacidades: Modelo de Capacidad y Madurez.

COTS: Son componentes que generalmente están relacionados con actividades que son repetidas con alta frecuencia en la mayoría de las áreas de proceso que se modelan y que una vez proyectados en el diseño arquitectónico, resultan funcionalidades de alta incidencia en la colaboración y de elevada frecuencia de solicitud en la ejecución del sistema.

Framework: Marco de trabajo.

IEEE: Es un organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional.

No COTS: Contienen funcionalidades de alta incidencia en la colaboración y de elevada frecuencia de solicitud en la ejecución del sistema.

RUP: *Rational Unified Process.* Compañía que lleva la delantera en lo referente a Arquitectura de Software.

SEI: *Software Engineering Institute.* Instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software, que dieran respuesta a los problemas que generaba al ejército estadounidense la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Financiado por el Departamento de Defensa de los Estados Unidos y administrado por la Universidad Carnegie Mellon.

UML: *Unified Modeling Language*. Lenguaje Unificado de Modelado.

Referencias Bibliográficas

1. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque prático.* s.l. : Connecticut , 2002.

2. **Bass, Len, Clements, Paul and Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison Wesley, 2003. 0-321-15495-9.

3. **Garlan, D. y Perry, D. E.** *Special Issue on Software Architecture.* s.l. : IEEE Transctions on Software Engineering, 1995.

4. **Garlan, D. y Perry, D. E.** *Advances in Software Engineering and Knowledge Engineering.* In V. Ambriola and G. Tortora (ed.), Series on Software Engineering and Knowledge Engineering, Vol 2, World Scientific Publishing Company, Singapore, pp. 1-39, 1993

5. **Kruchten, Philippe.** *The Rational Unified Process.* s.l. : Addison Wesley Longman, 2003.

6. **Bosch, J.** *Design & Use of Software Architectures*. . s.l. : Addison-Wesley, 2000.

7. **Shaw, M. & Garlan, D.** *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice-Hall, 2008.

8. **Buschman, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M.** *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Vol. 2. Estados Unidos, 2000.

9. **Clements, Paul, Kazman, Rick y Klein, Mark.** *Evaluating Software Architectures.* s.l. : SEI. Series in Software Engineering. Adisson Wesley, 2002.

10. **Rey, Billy Reynoso: Billy.** Carlos. [Online] 26 de 6 de 2006. [Citado el: 28 de 12 de 2007.]. Atlassian.com.

11. **Kruchten, Philippe.** *The 4+1 View Model of Architecture.* s.l. : IEEE Software, Noviembre de 1995. pp. pp. 42-50.

12. **Buschmann, F, y otros. 1996.** *Pattern – Oriented Software Architecture. A System of Patterns.* 1ra Edición. Inglaterra : John Wiley y Sons, 1996. pág. 666. ISSN: 978-0471606956 .

13. **Grady Booch, James Rumbaugh e Ivar Jacobson.** *El Lenguaje Unificado de Modelado. .* Madrid : Addison-Wesley, 1999.

14. *Gómez, Alvaro.* [Online] 3 de 9 de 2007. [Citado el: 15 de 4 de 2012.]. mundoenlinea.cl.

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

16. **2000, IEEE Std 1471-.** *Recommended Practice for Architectural Description for Software-Intensive Systems.* pág. 3.

17. **R, Pressman.** *Ingeniería del Software. Un enfoque práctico. Parte 1.* La Habana Cuba : Félix Varela, 2005.

18. **Nick Rozansk, Eóin Woods.** *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives (Hardcover).*

19. **Lazo, René.** *Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión.* 2011.

20. **Boehm, Barry y Turner, Richard.** *Guia de ingenieria del software.* 2011.

21. **Shaw, Mary y Clements, Paul.** *A field guide to Boxology: Preliminary classification of architectural styles for software systems. Computer Science Department and Software Engineering Institute.* s.l. : Carnegie Mellon University.

22. **Piñero, Pedro Y.** *Entrevista personal. Laboratorio de GESPRO.* 2012.

23. **Santana, Felix N. Abelardo.** *Entrevista personal. Laboratorio de GESPRO.* 2012.