Universidad de las Ciencias Informáticas

Facultad 3

Modelo de variabilidad para el sistema Xedro-ERP de la línea de producto Xedro.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autor: Zoíma Guerra Jardines

Tutor: Ing. Yusniel Matos Arias

Co-Tutor: MsC. Nemury Silega Martínez

Ing. Andrés Reynaldo Milord

Ciudad de La Habana

Junio 2014

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de Junio del año 2014.

Zoíma Guerra Jardines

_____
Firma del Autor

Ing. Yusniel Matos Arias

_____
Firma del Tutor

Agradecimientos

AGRADECIMIENTOS

A mis padres: Gracias por la oportunidad de existir, por su ejemplo de superación incasable, por su comprensión y confianza, por su amor, porque sin su apoyo no hubiera sido posible la culminación de mi carrera profesional.

A mis segundos padres: María e Ito por todo su apoyo, dedicación, por su amor incondicional.

A mi hermana querida Zoemi Guerra Jardines que es mi razón de ser.

A mis hermanos Wilfredo Palma y Arletis Francis, que aunque no de sangre si de corazón, no me olvidaré de ellos.

A mi novio: Por ayudarme siempre, por guiarme, por enseñarme que las cosas no cambian, cambiamos nosotros, por soportar mis majaderías y por estar ahí en cada uno de los momentos que lo necesité, gracias.

A mi familia: Por todo el apoyo brindado a través de mis estudios, por siempre confiar en mí, gracias.

A mi tutor y mis co-tutores: Por ser los más exigentes y obligarme continuamente a dar lo mejor de mí.

A mis amigas y amigos Victoria Sánchez, Dalilis Cisnero, Samuel A. Cruz, Eddy Núñez, Leonardo Arzuaga y Rodrigo gracias por todos los momentos compartidos: los alegres y los tristes, gracias por estar ahí para mí, siempre que lo he necesitado.

A todos mis amigos, los que están en UCI y los que están fuera de ella.

Agradezco a todas aquellas personas que de una forma u otra han estado presentes en mi vida, y con los cuales he compartido todo tipo de momentos.

DEDICATORIA

A mis padres (Milagros y Enrique Valentín) que guían cada uno de mis pasos, que me inspiran desde sus amorosas miradas y me hacen heredera de la mayor fortuna del mundo LA VIDA. Porque gracias a su cariño, guía y apoyo he llegado a realizar uno de los sueños más grandes de mi vida la culminación de mis estudios profesionales que constituyen el legado más grande que pudiera recibir y por lo cual les estaré eternamente agradecido, los quiero.

RESUMEN

Las Líneas de Producto Software pueden englobarse dentro de la reutilización de software en el proceso que propone un conjunto de métodos y técnicas para el desarrollo de varios productos que poseen un conjunto de características afines. La manera más común de representar todos los posibles productos de la misma línea de productos es mediante modelos de funcionalidades o características (modelos de variabilidad).

En este trabajo se presenta el proceso de definición de un modelo para representar la variabilidad de la línea de productos Xedro-ERP, del Centro de Informatización de la Gestión de Entidades.

La propuesta se sintetiza con un modelo de variabilidad de la línea de producto Xedro-ERP para contribuir a mejorar el análisis de esta cualidad. De esta manera, se promueve el desarrollo de soluciones en la Universidad de las Ciencias Informáticas, en aras de robustecer la calidad de los productos desarrollados.

Palabras clave: línea de producto, modelo, variabilidad.

TABLA DE CONTENIDO

Tabla de contenido

INTRODUCCIÓN

El desarrollo del *software* es complejo por la aparición de nuevas tecnologías, la interconexión de varios sistemas y plataformas, la adaptación personalizada del *software* a cada tipo de usuario, la necesidad de integrar viejos sistemas aún válidos. Estas circunstancias conllevan a múltiples versiones de la misma aplicación. Por ello, la Ingeniería del *Software* debe proporcionar herramientas y métodos que permitan desarrollar una familia de productos con distintas capacidades y adaptables a las necesidades del mercado, y no sólo un único producto. (Espinosa, y otros, 2011)

De lo anterior explicado surge el concepto de Línea de Producto de Software (LPS) donde (Clements, 2001) la define de la siguiente manera: “se definen las LPS como un conjunto de sistemas *software*, que comparten un conjunto común de características, las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base de una manera preestablecida”

(Bass, 2003), (Clements, 2002) reconocen que quizás el activo imprescindible en este conjunto es la arquitectura. Igualmente que “un componente clave para la solución efectiva de esos problemas es el uso de una arquitectura de línea de productos que permita a la organización identificar y reusar los artefactos. La arquitectura es, en ese sentido, la que asegura la cohesión de la línea de productos” (Zubrow).

La arquitectura tiene como objetivo fundamental la reutilización durante el desarrollo de software (Sinnema, 2004). La filosofía que persigue es explotar los elementos comunes entre sistemas software y al mismo tiempo mantener la habilidad de variar las funcionalidades (Sinnema, 2004). La cualidad anterior se conoce comúnmente, en el ámbito de las líneas de productos de software, como variabilidad (Sinnema, 2004). La representación y gestión de la variabilidad corresponde uno de los elementos claves para una LPS.

La facultad 3 cuenta con el Centro de Informatización de Gestión de Entidades (CEIGE), el cual persigue el enfoque de LPS orientado a Xedro. Xedro es una novedosa solución informática que se construye adaptado al control de los recursos empresariales de la entidad, conocidos internacionalmente como Sistemas ERP (Sistema Integral de Gestión CEDRUX, 2013).

Introducción

Los sistemas ERP, surgieron de la necesidad de englobar los datos referentes a la cadena de producción de las empresas, con el fin de brindar información confiable en tiempo real. ERP permite el acceso permanente a la información generada por cada área de la empresa, pudiendo acudir a los datos de manera segura y oportuna. (ERP, 2013)

Xedro-ERP tiene como fin desarrollar una familia de productos para la gestión de entidades presupuestadas y empresariales que se rigen por las normas establecidas en Cuba. También provee facilidades para la integración de las diferentes áreas productivas y departamentos administrativos.

Xedro-ERP busca lograr los beneficios de los modelos de desarrollo de software basados en la reutilización de componentes, sin embargo los productos de la línea son desarrollados por separado sin tener en cuenta los puntos en común haciéndose difícil la reutilización de cualquier componente de la línea. Este problema se refleja en el Expediente de proyecto de los productos de Xedro-ERP, documento ARQUITECTURA DE SOFTWARE. Donde se conoce que: Dentro de los subsistemas Banco, Caja, COPA no existe reutilización de componentes. Mientras que para Contabilidad, se conoce que: no se reutilizaron otros componentes o elementos externos en la solución del subsistema.

La línea está presentando conflicto con las personalizaciones a las soluciones debido a que no se tiene un previo análisis del dominio. Este análisis está enfocado con la creación de nuevos componentes o la actualización de los ya existentes, para evitar que se implemente una funcionalidad desarrollada con anterioridad o que no sea parte del dominio de la línea. Este problema está relacionado con el tiempo de desarrollo de las personalizaciones realizadas a la línea. En marzo del 2014, se inició el proceso de personalización al Grupo Empresarial de la Industria Ligera (GEIL). Donde se añadió una funcionalidad al módulo Contabilidad, que consiste en asignar un tipo de cuenta, detallando como se cerraría la cuenta nominal (cierre ejercicio o cierre período). De la misma manera en Estructura y Composición se adicionó en el nomenclador tipo de cifra, que los campos a llenar sean opcionales. Para cada una de estas funcionalidades se consideró un tiempo de una semana. Pero nunca se logró terminar esta personalización a GEIL pues fue cancelado por el cliente en abril, no siendo útil para la presente investigación.

En encuentros con los especialistas de la línea salió a relucir la necesidad de documentar las posibles variantes que podrían existir teniendo en cuenta los componentes desarrollados con anterioridad y buscar un mecanismo que permita controlar estas configuraciones.

Asumiendo que la variabilidad es la habilidad que tiene un producto para ser cambiado y a la vez es un mecanismo que permite ser configurado para usarse en varios contextos.

El desconocimiento de variabilidad que presenta la línea de productos Xedro-ERP dificulta el análisis del dominio en la toma de decisiones posteriores. De lo anterior planteado, se formula como **Problema a Resolver:** la dificultad del análisis de variabilidad en la línea de producto Xedro-ERP aumenta el tiempo de desarrollo de las nuevas personalizaciones.

La investigación tendrá como **Objeto de estudio:** la variabilidad en líneas de productos de software. En esta investigación se definió como **Campo de acción:** la variabilidad en la línea de producto Xedro-ERP.

Para dar solución al problema se ha planteado como **Objetivo general:** definir un modelo de variabilidad para la línea de productos Xedro-ERP, que facilite su análisis y mejore el tiempo de desarrollo de las nuevas personalizaciones. Como **Idea a defender:** se tiene que, la definición y aplicación de un modelo que permita el análisis de variabilidad reduciría a Xedro-ERP el tiempo de desarrollo de las nuevas personalizaciones realizadas a la línea. Para lograr el objetivo propuesto se acometen los siguientes **Objetivos específicos:**

- Elaborar el marco teórico de la investigación que permita identificar los principales elementos de variabilidad de una LPS.
- Desarrollar y aplicar el modelo de variabilidad en la línea de productos Xedro-ERP.
- Validar la propuesta de solución.

Posibles resultados:

El estudio de esta investigación planea resultados placenteros contribuyendo al completo dominio y solución del problema planteado. Quedarán reunidos todos los parámetros necesarios para la confección de un modelo de variabilidad que permita a cualquier LPS tomar decisiones del análisis del dominio. Con el transcurso de la investigación se presenta la variabilidad de la línea de productos Xedro-ERP. Esta variabilidad permite la integración de

nuevos productos, la realización de nuevas personalizaciones y la construcción de nuevas configuraciones a la LPS. Además de lograr menor tiempo de desarrollo de las soluciones. De esta manera, se promueve el desarrollo de soluciones en la UCI, en aras de robustecer la calidad de los productos desarrollados.

Nota: Para más información consultar documento Modelo de variabilidad y variabilidad en la siguiente dirección:

- CIG-XDO-N-1-Modelo_Variabilidad
- CIG-XDO-N-2-Variabilidad_Xedro

Aporte:

- Modelo de variabilidad que permita el análisis del dominio de una LPS.
- Variabilidad de la línea de productos Xedro-ERP.

El presente trabajo consta de 3 capítulos que están divididos en epígrafes y sub-epígrafes.

Capítulo 1. Fundamentación Teórica: Se analizan conceptos relacionados con la investigación, estableciendo de este modo el inicio de la misma. Se define un marco de trabajo delimitado por el objeto de estudio y la situación problemática, donde se examinan las posibles vías de solución del problema a resolver.

Capítulo 2. Modelo de variabilidad de Xedro-ERP: En este capítulo se muestra la propuesta de solución que incluye el modelo de variabilidad con la descripción textual correspondiente al mismo. Se detalla todo el proceso de construcción del modelo de variabilidad para la línea Xedro-ERP.

Capítulo 3. Validación de la solución: Se realizan las pruebas necesarias para verificar los resultados alcanzados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se presentará la fundamentación teórica de la investigación a través de la realización de un estudio sobre temas relacionados con las LPS; lo referente a la variabilidad de una LPS, los modelos de variabilidad así como el estado del arte de estos modelos. También se iniciará el estudio de FAMA, *framework* que permite el análisis automático del modelado de variabilidad en una LPS.

1.2. Línea de Producto de Software

Las LPS pueden englobarse dentro de la reutilización como aspecto clave para reducir los costos, aumentar la productividad y asegurar la calidad de los productos. (Montilva, 2006) caracteriza la reutilización en las LPS por ser: estratégica y gestionarse de forma sistemática.

“La reutilización de *software* es el proceso de implementar o actualizar sistemas de *software* usando activos de *software* existentes”. (Sodhi, 1999)

“Reutilización de *software* es el proceso de crear sistemas de *software* a partir de *software* existentes, en lugar de desarrollarlo desde el comienzo”. (Montilva, 2006)

Para (Krueger, 2006) y (Tussén, y otros, 2013) la reutilización consiste en desarrollar elementos de *software* que puedan utilizarse más de una vez con la mínima cantidad de modificaciones, garantizando que al reutilizar un elemento de *software*, libre de defectos, implicará que el sistema que lo utilice no tendrá problema alguno en lo que respecta a dicho elemento.

(Brooks., 1987) indica que la reutilización de código en sí no es la ventaja de una LPS, sino la reutilización de decisiones hechas mucho antes de la fase de programación. En las LPS se promueve la reutilización al extremo en el cual más que reutilizar componentes o artefactos se reutilizan decisiones de cómo afrontar la variabilidad de los productos en varios aspectos claves (Northrop., 2007a) (Jones., 2008).

Para (Díaz, y otros, 2010) la definición más aceptada de una LPS procede de (Clements, 2001) donde “se definen las líneas del producto de *software* como un conjunto de sistemas *software*, que comparten un conjunto común de características, las cuales satisfacen las

necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base de una manera preestablecida".

En esta definición se pueden detectar cinco aspectos fundamentales para identificar una LPS (Díaz, y otros, 2010):

- Un conjunto de sistemas *software*. Se hace referencia al desarrollo de un conjunto de productos, no al desarrollo de un producto; y delimita el alcance de este conjunto.
- Conjunto común de características. Este conjunto de productos poseen características afines. Una característica es una peculiaridad del producto que los clientes consideran importante para describir y distinguir entre los distintos miembros de la LPS.
- Satisfacen las necesidades de un segmento particular de mercado. Una LPS se desarrolla orientándose a un segmento de mercado concreto, es decir, los productos pretenden satisfacer las necesidades de un segmento de mercado. De la habilidad para identificar este mercado, dependerá el éxito de la LPS.
- Se desarrollan a partir de un sistema común de activos base. Los productos son desarrollados a partir de un conjunto común de activos reutilizables. Este término engloba elementos, tales como requisitos, arquitecturas, componentes, código fuente, etc., que conforman la base sobre la que se construye el producto.
- Manera preestablecida. Los productos se construyen de una forma preestablecida, la estrategia para construir el producto está establecida a través de un plan de producción.

1.2.1. Modelo básico de una LPS

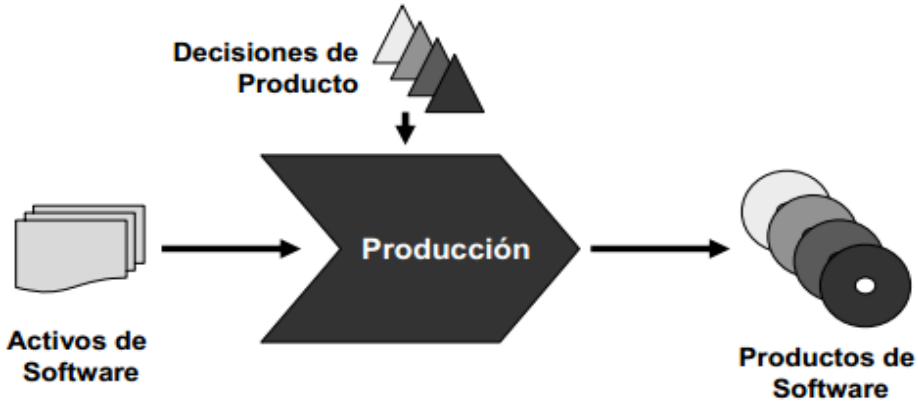


Figura 1 Modelo básico de una LPS (Montilva, 2006).

Este modelo está compuesto por la entrada, el control, el proceso de producción y la salida (Montilva, 2006).

La entrada: Activos de software.

Es una colección de partes de software (requisitos, diseños, componentes, casos de prueba, etc.) que se configuran y componen de una manera prescrita para producir los productos de la línea.

El control: Decisiones de producto.

Los Modelos de Decisiones describen los aspectos variables y opcionales de los productos de la línea. Cada producto de la línea es definido por un conjunto de decisiones (decisiones del producto).

El proceso de producción.

Establece los mecanismos o pasos para componer y configurar productos a partir de los activos de entrada; además las decisiones del producto se usan para determinar qué activos de entrada utilizar y cómo configurar los puntos de variación de esos activos.

La salida: Productos de software.

Conjunto de todos los productos que pueden o son producidos por la línea de productos.

1.2.2. Proceso de desarrollo

El proceso de desarrollo de una LPS no intenta construir una aplicación, sino una familia de ellas. Este enfoque supone un cambio de un desarrollo orientado a un único producto software, a un desarrollo de varios productos que contienen unas características comunes, formando una familia de productos (Sánchez, 2011). Siguiendo el enfoque propuesto por (Pohl K., 2005), el paradigma Ingeniería de Líneas de Producto Software (ILPS) está dividido en dos procesos: La ingeniería de dominio y la ingeniería de aplicación. La Figura 2 muestra el proceso total de los dos enfoques en la que se observa el doble ciclo de vida aplicado a las LPS.

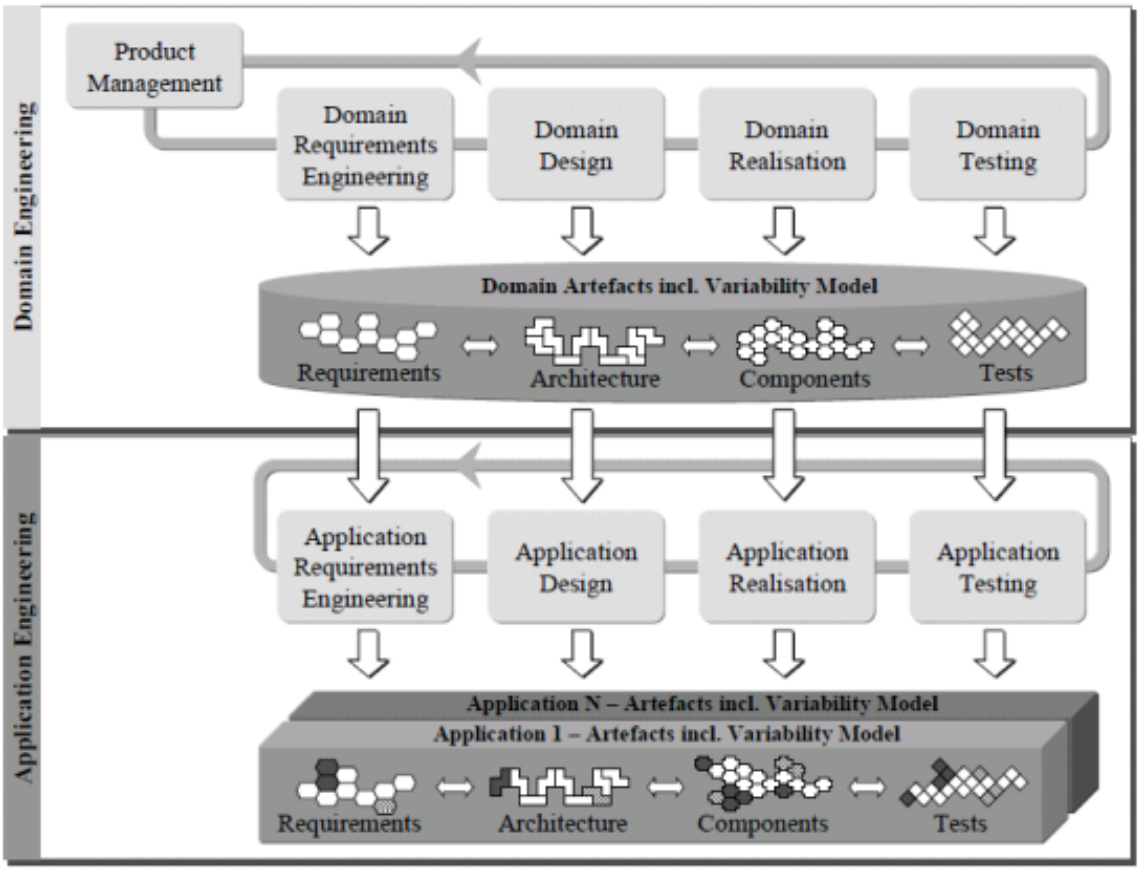


Figura 2 Enfoque del doble ciclo de vida aplicado a las líneas de producto (Pohl K., 2005).

La Ingeniería de Dominio se centra básicamente en el estudio del dominio con el objetivo de identificar las partes comunes y variables de la familia. Se encarga también de la gestión de la variabilidad en el desarrollo de los activos base y en el diseño del plan de producción (Sánchez, 2011). Este proceso se define como:

"La ingeniería de dominio es el proceso de las ILPS en el cual la parte común y la variabilidad de la línea de producto son definidas y realizadas" (Pohl K., 2005).

En la Figura 2 muestra que este proceso tiene 5 subprocesos formando un ciclo circular, la salida que producen los subprocesos son la entrada para otros. Los subprocesos son los siguientes (Pohl K., 2005):

- Gestión de productos: Encargado de los aspectos económicos y de la estrategia de mercado. Define el alcance de la LPS. Este proceso tiene como entrada los objetivos de la compañía y produce como salida las principales características comunes y

variables de la LPS, una planificación de las entregas y una lista de los productos existentes y/o artefactos a reutilizar.

- Ingeniería de requisitos de dominio: Su principal tarea es la documentación y elicitación de requisitos comunes y variables de la línea de producto. Tiene como entrada la hoja del producto, es decir, las principales características comunes y variables del producto. Produce como salida los modelos de requisitos, reutilizables y textuales, un modelo de variabilidad de la línea de producto.
- Diseño de dominio: En este subproceso se define la arquitectura de referencia de la LPS. Recibe como entrada los requisitos del dominio y su modelo correspondiente. Produce como salida la arquitectura de referencia y un modelo de variabilidad ya refinado.
- Realización de dominio: En esta fase es donde se realiza el diseño e implementación de componentes *software* reutilizables. Recibe como entrada la arquitectura de referencia y la lista de artefactos reutilizables que se han de desarrollar. Como salida produce el diseño e implementación de componentes *software* reutilizables.
- Pruebas de dominio: En esta fase se lleva a cabo la validación y verificación de componentes *software* reutilizables. Recibe como entrada los requisitos de dominio, la arquitectura de referencia y el diseño e implementación de componentes reutilizables. Como salida produce el resultado de los test.

En este enfoque de desarrollo intervienen los siguientes artefactos de dominio (Pohl K., 2005):

- Mapa de ruta de la línea de producto: Describe las características de todas las aplicaciones de la LPS y clasifica las características en características comunes que forman parte de cada aplicación y características variables sólo forman parte de algunas aplicaciones.
- Modelo de variabilidad del dominio: Define la variabilidad de la LPS.
- Requisitos del dominio: Enmarca los requisitos que son comunes a todas las aplicaciones de la LPS así como los requisitos variables que permiten la derivación de los requisitos personalizados para las diferentes aplicaciones.

- Arquitectura de dominio: La arquitectura de dominio o arquitectura de referencia determina la estructura y la textura de las aplicaciones en la LPS. La estructura determina la descomposición estática y dinámica que es válida para todas las aplicaciones de la línea de productos. La textura es el conjunto de reglas comunes que guían el diseño y la implementación de las partes y cómo éstas son combinadas para formar las aplicaciones.
- Artefactos de implementación de dominio: Se encargan del diseño e implementación de los artefactos de los componentes e interfaces *software* reutilizables. El diseño de artefactos abarca los diferentes tipos de modelos que capturan la estructura estática y dinámica de cada componente. La implementación de artefactos incluye archivos de código fuente, archivos de configuración, etc.
- Artefactos de test de dominio: Los artefactos de test de dominio incluyen el plan de pruebas de dominio, los casos de prueba de dominio y los escenarios de prueba de dominio. El plan de prueba de dominio define la estrategia de prueba para probar el dominio, los artefactos de prueba que son creados, y los casos de prueba que son ejecutados. Los casos de prueba y los escenarios de prueba de dominio proporcionan instrucciones detalladas para el ingeniero de pruebas que realiza las pruebas.

La Ingeniería de Aplicación se centra básicamente en el desarrollo de productos concretos, en obtener una aplicación en concreto reutilizando la mayor cantidad posible de artefactos del dominio y en explotar la parte común y variable de la LPS durante el desarrollo de la aplicación (Sánchez, 2011). Este proceso se puede definir como:

“La ingeniería de aplicación es el proceso de las ILPS en el cual las aplicaciones de la línea de producto se construyen reutilizando artefactos de dominio y explotando la variabilidad de la familia de producto.” (Pohl K., 2005).

En la Figura 2 se muestra que el proceso de Ingeniería de Dominio consta de 4 subprocesos formando un ciclo circular, la salida que producen los subprocesos son la entrada para otros. Los subprocesos que forman el proceso de Ingeniería de Dominio son los siguientes (Pohl K., 2005):

- Ingeniería de requisitos de aplicación: Su principal tarea es la especificación de los requisitos de la aplicación y del producto. Tiene como entrada la hoja del producto (principales características comunes y variables del producto), requisitos del dominio y

requisitos adicionales del cliente. Produce como salida la especificación de requisitos de una aplicación en particular.

- Diseño de aplicación: En esta etapa se procede a la construcción de la arquitectura de la aplicación mediante la derivación de la arquitectura de referencia. Recibe como entrada la especificación de requisitos de la aplicación y la arquitectura de referencia. Produce como salida la arquitectura de la aplicación.
- Realización de la aplicación: En esta fase es donde se realiza el desarrollo de la aplicación, la selección y configuración de componentes reutilizables y la derivación de los artefactos. Recibe como entrada la arquitectura de la aplicación y el diseño de los artefactos de la aplicación. Como salida produce la aplicación ejecutable.
- Pruebas de aplicación: En esta fase se lleva a cabo la validación y verificación de la aplicación. Recibe como entrada los requisitos de la aplicación, la arquitectura de la aplicación, el diseño e implementación de las componentes de la aplicación, la aplicación resultante y los artefactos de test reutilizables generados en el subproceso *domain testing*. Como salida produce un informe detallado del proceso de prueba.
- Modelo de variabilidad de aplicación: Documentan para una aplicación en particular los enlaces de la variabilidad junto con los fundamentos para seleccionar estos enlaces.
- Requisitos de aplicación: Constituyen la especificación completa de requisitos de una aplicación concreta.
- Arquitectura de aplicación: La arquitectura de aplicación determina la estructura general de una aplicación determinada. Es una instancia específica de la arquitectura de referencia. Para el éxito de la línea de producto, es esencial reutilizar la arquitectura de referencia en todas las aplicaciones.
- Artefactos de implementación de aplicación: Engloban el diseño de componentes e interfaces de una aplicación específica, así como su configuración y ejecución de la aplicación.
- Artefactos de prueba de aplicación: Comprenden la documentación de prueba para una aplicación específica.

Como se dijo anteriormente, una LPS está compuesta por un conjunto de productos con características afines. Donde cada uno de estos productos está construido a partir de un

conjunto de activos reutilizables, la cual es desarrollada para un dominio determinado. Una LPS no intenta construir una aplicación, sino una familia de ellas. La línea de producto Xedro-ERP persigue este enfoque. Este enfoque se basa en el desarrollo de varios productos que poseen características comunes, integrando una familia de productos. El presente trabajo utiliza para su desarrollo estos conocimientos pues están relacionados con el objetivo primario de esta investigación, que es, representar la variabilidad de la línea de producto Xedro-ERP. El siguiente epígrafe abordará diferentes temas relacionados con la variabilidad de una LPS.

1.3. Variabilidad en LPS

La representación y la gestión de la parte común y variable de una LPS corresponde una de las principales actividades de su desarrollo, junto con la configuración y la derivación de los productos finales. La variabilidad es el elemento más característico de las LPS. Su gestión eficaz es principal para definir la línea de producto satisfactoriamente (Sánchez, 2011). La variabilidad se define como:

“La variabilidad es la habilidad de un sistema *software* o artefacto para ser cambiado, personalizado o configurado para usarse en múltiples contextos” (Van Group J., 2002).

Algunos conceptos básicos relacionados con el concepto de variabilidad se definen a continuación:

“La variabilidad sujeto es un elemento variable del mundo real o una propiedad variable del propio elemento” (Pohl K., 2005).

“La variabilidad objeto es una instancia particular de la variabilidad sujeto” (Pohl K., 2005).

“Un punto de variabilidad es una representación de una variabilidad sujeto dentro de los artefactos de dominio enriquecidos por información contextual.” (Pohl K., 2005).

“Una variante es una representación de una variabilidad objeto dentro de los artefactos de dominio” (Pohl K., 2005). Es una entidad diferente en requisitos, arquitectura, etc., que identifica una única opción de un punto de variabilidad.

Los puntos de variabilidad y variantes son utilizados para definir la variabilidad de la LPS. El proceso para especificar la variabilidad es el siguiente: (Sánchez, 2011)

- Identificación del sujeto de variación, es decir, identificación del elemento que varía en el mundo real.

- Definición del punto de variabilidad en el contexto de la LPS.
- Definición de las variantes, es decir, selección de los objetos de variabilidad de LPS y la definición de los objetos de variabilidad seleccionados como variantes.

Para gestionar la variabilidad en una línea de producto, se necesita diferenciar tres tipos principales de variabilidad: (Sánchez, 2011)

- Opcional: Es aquella en la que la variante de un punto de variabilidad puede ser parte o no de la línea de producto de una aplicación. Establece relación entre el punto de variabilidad y la variante con una cardinalidad mínima de 0 y una cardinalidad máxima de 1.
- Obligatoria: La variante de un punto de variabilidad deber ser parte de la línea de producto de una aplicación. Establece una relación entre el punto de variabilidad y la variante con una cardinalidad mínima de 1 y una cardinalidad máxima de1.
- Alternativa múltiple: Se da cuando un punto de variabilidad puede seleccionar o no más de una variante. Establece relación entre el punto de variabilidad y la variante con la cardinalidad mínima entre 0..1, y máxima de N.

En el proceso de Ingeniería de Dominio se define la variabilidad de una LPS mediante un modelo de variabilidad que captura la variabilidad del dominio en requisitos, arquitectura, componentes y pruebas, y se explota en la etapa de Ingeniería de Aplicación (Sánchez, 2011).

Según la (RAE) un modelo es un “arquetipo o punto de referencia para imitarlo o reproducirlo; es un esquema teórico,… de un sistema o de una realidad compleja…, y es la representación en pequeño de alguna cosa”. (Silvana, y otros, 2012) y (Wilson, 1993) define un modelo mediante “… la interpretación explícita de lo que uno entiende de una situación, o tan solo de las ideas de uno acerca de esa situación. Puede expresarse en símbolos o palabras pero en esencia es una descripción de entidades, procesos o atributos y las relaciones entre ellos”. En esta investigación se modelará la variabilidad de una LPS a través de un Modelo de variabilidad, dando apertura al siguiente tema de investigación.

1.3.1. Modelado de la variabilidad en LPS

Con el desarrollo de una LPS es posible construir, de forma rápida y eficiente, múltiples variaciones de un mismo producto. La variabilidad es un concepto central en el desarrollo de

LPS. La misma posibilita el reuso constructivo y facilita la derivación de diferentes productos específicos para cada cliente de la línea de producto (Pohl., 2003). En la ILPS es necesario que existan métodos de modelado que expresen las características comunes y las variables entre los productos de una línea. Dicha tarea se define como modelado de la variabilidad y esta es la propiedad que diferencia el desarrollo de líneas de producto del desarrollo de *software* tradicional (Pohl K., 2005).

Los modelos de variabilidad que se generan en el desarrollo de una LPS pueden tener gran cantidad de características y un gran número de combinaciones entre ellas. El análisis de los modelos de variabilidad, permiten extraer automáticamente informaciones importantes de los modelos, las cuales pueden ser útiles a los analistas, diseñadores, programadores y gestores de la línea de productos.

1.3.2. Modelo de Variabilidad

Para tratar la variabilidad han sido realizadas y discutidas diferentes aproximaciones en los últimos años. Sin embargo, actualmente no existe una forma estándar para representarla (Espinosa, y otros, 2011). Muchas de las aproximaciones propuestas agregan anotaciones de variabilidad directamente en los modelos que captan la funcionalidad del software, mientras que otras aproximaciones separan la especificación de la variabilidad en modelos específicos (Espinosa, y otros, 2011).

Estos modelos de variabilidad pueden ser plasmados por modelos de características (Espinosa, y otros, 2011). Sin embargo, (Sánchez, 2011) alega que la variabilidad de una LPS se puede detallar además a nivel de arquitectura y a través de la especificación de los componentes.

Los modelos de características representan gráficamente una línea de producto, a través de las posibles combinaciones entre las características. El conjunto de características se organizan de la siguiente forma:

- Una relación entre característica padre y característica hija.
- Relaciones no jerárquicas que suelen ser del tipo: si la característica A aparece, entonces las características B se debe incluir (o excluir).

Modelos de características básicos: son aquellos modelos en los que se usan relaciones simples entre las características.

Desde antes de existir el concepto de LPS existía la herramienta de análisis de dominio (AD), que se enfoca en analizar los aspectos comunes de un dominio particular que podría consistir en un conjunto de sistemas relacionados. A partir de esto, surgieron muchos métodos de AD, siendo el FODA el que marcó una pauta en el desarrollo del AD.

El primer modelo de características fue propuesto por (Kang, 1990) como parte del método de análisis FODA. Ha sido utilizado para extraer las similitudes y variabilidades de un dominio. Su principal objetivo es la identificación de características relevantes de sistemas de *software* que pertenecen a un dominio. Está compuesto por tres fases, para las cuales hay una serie de actividades individuales, la más relevante es la del modelado de características, pues es la que define por medio de un árbol donde quedarán especificados los elementos comunes y variables del dominio. En FODA un modelo está compuesto de dos elementos: las características y las relaciones entre ellas. Las características están organizadas en una estructura jerárquica en forma de árbol. Una de las características del árbol es la raíz y representa el sistema como un todo.

Más tarde (Kang, 1990) propuso FORM como extensión de FODA. En esta propuesta sus autores consideraron la importancia del uso de los modelos de características para modelar la variabilidad en otras fases del desarrollo además de la ingeniería de requisitos. En (Griss. M., 1998) presentaron Feature-RSEB, una extensión de FODA que presenta una nueva notación gráfica para el modelo, pero que no cambia su semántica. Finalmente (Czarnecki K., 2005) propuso otra extensión de FODA: el *cardinality-based Feature models*.

A partir de lo anterior, la variabilidad en LPS describe las diferencias entre los sistemas que pertenecen a una línea de producto y representan los aspectos que pueden cambiar en los productos de la línea.

Luego del análisis, se percibió la importancia de la variabilidad para una LPS, ya que la representación de la variabilidad pertenece a una de las principales actividades para el desarrollo de una LPS. Como se dijo anteriormente, para representar la variabilidad han sido discutidas diferentes aproximaciones. Estas aproximaciones pueden ser plasmadas por modelos de características, a nivel de arquitectura y de componentes. Por tal razón, se culminó hacer un modelo de variabilidad que reúna estos aspectos. Estos aspectos están relacionados con el diseño arquitectónico, con el comportamiento de una LPS y también con la descripción de los componentes. Aspectos idóneos para modelar la variabilidad de la línea de producto Xedro-ERP.

1.4. Framework para el análisis de la variabilidad

La variabilidad en LPS es cada vez mayor, los modelos de variabilidad pueden tener miles de características. Por lo tanto, un apoyo automático se hace necesario para tratar dichos modelos.

Análisis Automáticos de Modelos de Características (siglas en inglés FAMA) (Trinidad, y otros, 2008) está desarrollado por un equipo de la Universidad de Sevilla. Permite el análisis automatizado de los modelos de características integrando algunos resolutores más comúnmente propuestos (BDD, SAT, CSP), siendo esta una característica poco común. Dispone de un *plugin* de Eclipse gráfico desarrollado en EMF bajo la licencia Open-Source. FAMA está trabajando conjuntamente con pure::Variants para integrar sus herramientas. Entre las ventajas que brinda el *framework* FAMA se encuentran su base formal, que garantiza una semántica fiable; su abstracción que permite el análisis de modelos distintos de los modelos de características; su capacidad de analizar modelos de características extendidos, en lo que se incluyen atributos; y el uso de varios resolutores distintos en su implementación. Por el contrario, las restricciones que permite son básicas (solamente implicación y exclusión). FAMA *Framework* (FAMA-FW) (Benavides, 2008) se ha desarrollado con el propósito de que otras personas integren sus técnicas de razonamiento automático.

FAMA es fácil de usar, tiene un simple y estable *QuestionTrader front-end*. Además de ser fácil de ampliar, la arquitectura FAMA FW permite ampliar o actualizar los productos existentes con sólo añadir o actualizar sus componentes o características. Terceras partes pueden desarrollar e integrar sus propias extensiones de FAMA. También permite la configuración fácil, FAMA FW está configurado por medio de un archivo XML única, facilitando su mantenimiento y configuración para adaptar la herramienta a las necesidades de los usuarios. Para su descarga se puede acceder al sitio en el cual aparecen todas sus versiones (FAMA, 2010).

El proceso general para automatizar el análisis de LPS empleando FAMA FW se esboza en la (Figura 3). Inicialmente, el modelo que describe la línea de productos es traducido para una representación lógica. Luego, se usan los resolutores para ejecutar las operaciones de análisis en la representación lógica del modelo. Para finalizar se obtienen los resultados contenidos en el modelo (Carneiro, 2011).

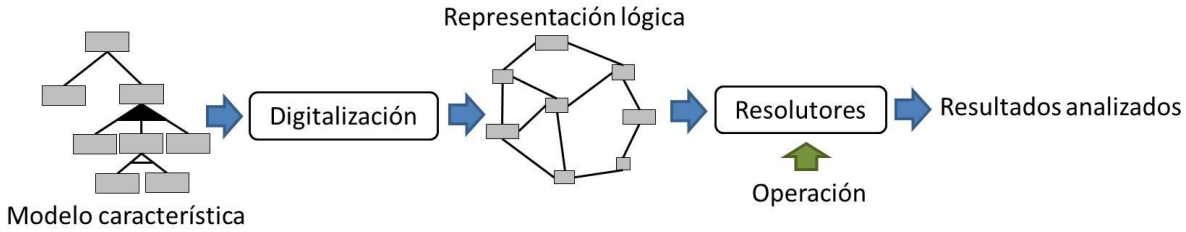


Figura 3 Proceso para el análisis automático de LPS (Carneiro, 2011).

Entre las operaciones ejecutadas por FAMA podemos encontrar:

- Determinan si un producto es válido para una LPS. Esta operación toma como entrada un modelo característico y un producto (un conjunto de características) y retorna un valor determinando si el producto pertenece al modelo o no.
- Determinan si un modelo es válido. Esta operación toma como entrada un modelo y retorna un valor determinando si tal modelo es válido o no. Un modelo es válido cuando representa por lo menos un producto. Un modelo de característica solo puede ser inválido cuando se hace el uso incorrecto de las relaciones no jerárquicas entre las características, generando de esta manera contradicciones en el modelo.
- Obtienen todos los posibles productos. Esta operación toma como entrada un modelo y retorna todos los productos válidos representados por dicho modelo.
- Obtiene las características básicas. Esta operación toma como entrada un modelo y retorna el conjunto de características que aparecen en todos los productos.
- Obtiene las características variantes. Esta operación toma como entrada un modelo y retorna el conjunto de características que no aparecen en todos los productos.

1.5. Conclusiones

El modelo de variabilidad de una LPS corresponde uno de las temáticas más importantes en este capítulo. Este modelo representa los aspectos comunes y variables (puntos de variación y variantes) de una LPS, el cual puede ser útil a los analistas, diseñadores, programadores y gestores de la línea de productos.

El estudio de la literatura especializada permitió conocer que las vías más utilizadas para representar la variabilidad es: mediante modelos de características (FODA, FORM, Feature-RSEB, entre otros). Sin embargo los autores reconocen que la variabilidad de una LPS se puede detallar además a nivel de arquitectura y a nivel de componentes. Por tal motivo, se

propone definir un modelo que esté compuesto por varias aproximaciones capaces de modelar una LPS por diferentes vías para así representar la mayor variabilidad posible dentro de una LPS.

CAPÍTULO 2: MODELO DE VARIABILIDAD DE XEDRO-ERP

2.1. Introducción

En este capítulo se propone un modelo de variabilidad que reduce el tiempo de desarrollo de las nuevas personalizaciones realizadas a la línea Xedro-ERP. Este modelo de variabilidad parte de la representación a nivel de arquitectura, la especificación de los componentes y mediante la representación de un árbol jerárquico compuesto por características y las relaciones existentes entre ellas.

2.2. Modelo conceptual de variabilidad

Este modelo de variabilidad está compuesto por 3 dimensiones: Variabilidad en arquitectura, Variabilidad en componentes y Variabilidad en características. Cada una de ellas tiene sus particularidades y se encargan de representar la variabilidad de una LPS. A continuación, se explica con más detalle cada una de estas dimensiones.

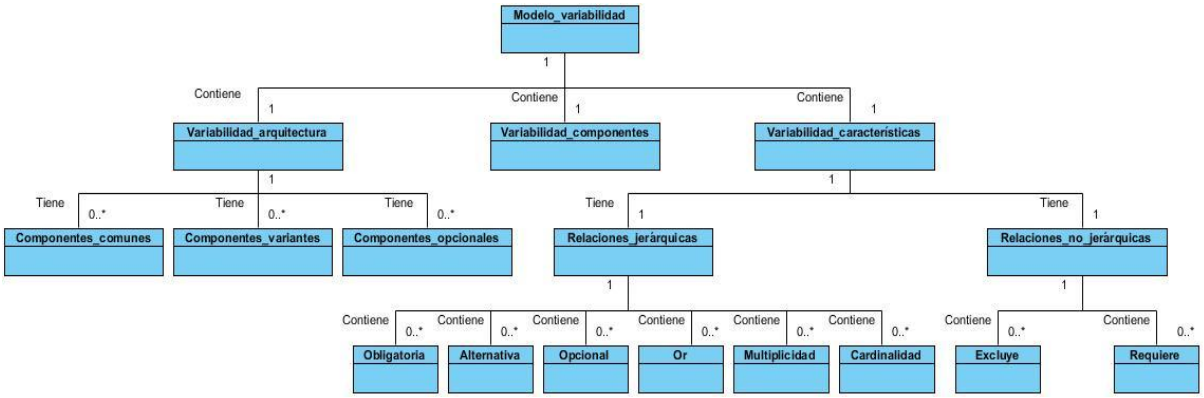


Figura 4 Vista general del Modelo de variabilidad

2.2.1. Variabilidad en Arquitectura

Para los sistemas complejos las arquitecturas de software corresponden el eslabón principal del proceso de diseño software. Las arquitecturas de software son el puente que permiten unir requisitos e implementación (Hofmeister, 2000).

La variabilidad a nivel de arquitectura se adquiere mediante los aspectos comunes y los aspectos variables de la arquitectura que son capturados por los componentes de *software* que son comunes a toda la familia, por los componentes de *software* que varían entre los miembros de la familia y los componentes opcionales.

Existen tres componentes para especificar la variabilidad en la arquitectura. Para entender estos componentes, a continuación se detallan cada uno de ellos:

- Componentes opcionales que son requeridos por algunos miembros de la familia de productos.
- Componentes comunes a todos los miembros de la familia de productos.
- Componentes variantes de los cuales algunos miembros de la familia de productos emplean distintas versiones.

Para confrontar estos componentes es necesario analizar las relaciones entre los componentes. Las relaciones entre componentes se realizan a través de interfaces (Jiménez Méndez). Las relaciones o conexiones entre componentes pueden ser:

- Interfaz. Una interfaz contiene una colección de operaciones y se utiliza para especificar los servicios de una clase o de un componente (Figura 5). La conexión de los componentes se realiza enlazando la interfaz requerida de un componente con la interfaz provista de otro.

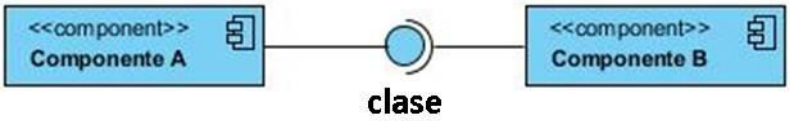


Figura 5 Relación con interfaces.

- Dependencia. Las relaciones de dependencia se utilizan para indicar que un componente se refiere a los servicios ofrecidos por otro componente (Figura 6).

Figura 6 Relación de dependencia.

- Asociación. Relación de dependencia que puede existir entre distintos componentes. Estas pueden especificarse como: Utiliza, Tiene, Contiene, Emplea y Genera (Figura 7).

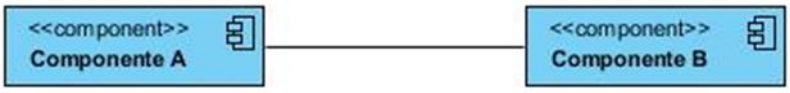


Figura 7 Relación de asociación.

2.2.2. Variabilidad en Componentes

Un componente de *software* es un elemento de un sistema *software* que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces definidas. Un componente define su comportamiento en términos de interfaces proveídas y requerida.

En ocasiones, parte del comportamiento de los componentes es común a toda la LPS y parte de su comportamiento cambia dependiendo del producto. Debido a esto, se especifica la variabilidad interna, es decir, la variabilidad a nivel de componentes se consigue mediante la especificación dentro de los componentes.

(Andrews, y otros, 2002) plantea la posibilidad de entender un componente por medio de tres perspectivas, estas son:

- Dominio: Representa en términos generales todos los aspectos del problema del usuario relacionado de forma directa con la funcionalidad que apoya el componente.
- Programa: Este enfoque es el que más varía de componente a componente, ya que muestra en forma más detallada la información técnica del componente como la estructura de los archivos de información, definiciones de las bases de datos, la definición de la interface de datos, los tipos de parámetros, información acerca de la invocación de los métodos del componente, etc.
- Situación: En este punto “El conocimiento del diseño y comportamiento de las entidades necesita ser especificado” (Andrews, y otros, 2002), la información que se determina en este punto es útil para igualar los requerimientos con las capacidades funcionales y no funcionales del componente; aquí se deben determinar aspectos como: estructura de la arquitectura física y conceptual, flujos de información que el componente transfiere, usa y transforma y tipos de algoritmos.

Luego de un análisis se especificará los componentes de la línea mediante las perspectivas que plantea (Andrews, y otros, 2002). Estas perspectivas recogerán una descripción basada en el objetivo y principales características del componente relacionado de forma directa con la funcionalidad que apoya el componente; los servicios que provee ese componente con

sus detalles, es decir, muestra de forma más detallada la información técnica del componente. Además se especificarán aspectos como flujos de información que el componente transfiere, usa y transforma. De esta forma se entenderá con claridad todo lo referente a la naturaleza de cada componente y así ayudara a entender con claridad su objetivo.

2.2.3. Variabilidad en características

La variabilidad entre productos de una LPS puede ser expresada en términos de características (Kang K., 1998), (Bosch, 2000). Una característica es una propiedad que representa una parte común o variable de un producto (Sánchez, 2011). Adecuadamente una característica se define como:

"Una característica es una unidad lógica de comportamiento que es especificada por un conjunto de requisitos funcionales o de calidad." (Bosch, 2000)

En esta definición se detectan dos aspectos fundamentales para identificar una característica:

- Unidad lógica de comportamiento. Manera lógica de proceder un sistema en relación con sus funcionalidades, es decir todo lo que el sistema sea capaz de manifestar (salida) o de ingresar (entrada).
- Requisitos funcionales. Un requisito funcional define una función del sistema de *software* o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales establecen el comportamiento del sistema.

La variabilidad a nivel de características se representa por medio de un árbol jerárquico compuesto por características y relaciones entre ellas. Las relaciones entre características pueden ser de dos tipos:

- Relación jerárquica. Esta relación es definida entre una característica padre y sus características hijas. Una característica hija solamente puede hacer parte de los productos en los que la característica padre aparece. Los tipos de relaciones jerárquicas son las siguientes:
  - Obligatoria: indica que cuando la característica padre hace parte de un producto particular, la característica hija también debe hacer parte del producto (Figura 8).

- Opcional: indica que cuando la característica padre hace parte de un producto particular, la característica hija, puede o no, ser incluida en el producto (Figura 8).
- Alternativa: es la relación entre una característica padre y un conjunto de características hijas que indica que cuando la característica padre hace parte de un producto particular, sólo una de las características del grupo de hijas debe hacer parte del producto (Figura 8).

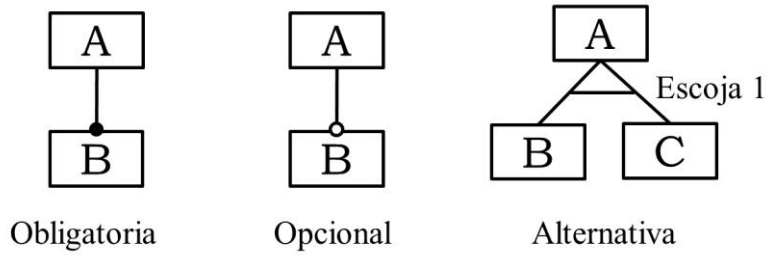


Figura 8 Relación jerárquica entre características.

- OR: indica que cuando la característica padre hace parte de un producto particular, una o más de las características del grupo de hijas debe hacer parte del producto (Figura 9).

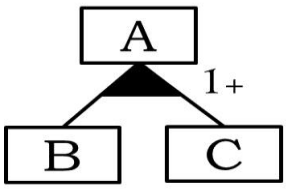


Figura 9 Relación jerárquica OR.

- Relación de grupo: indica que cuando la característica padre hace parte de un producto particular, el número de características hijas que debe hacer parte del producto depende de la multiplicidad. La multiplicidad equivalente a la relación alternativa es <1 – 1>, lo que significa que sólo una de las hijas del grupo pueden hacer parte del producto (Figura 10 (a)). La multiplicidad equivalente a la relación OR es <1 – n>, siendo n el número de características del grupo, lo que significa que una o n de las características del grupo pueden hacer parte del producto (Figura 10 (b)).

- Relación con cardinalidad: indica que cuando la característica padre hace parte de un producto particular, la inclusión de la característica hija depende de la cardinalidad. La cardinalidad equivalente a la relación obligatoria es [1..1], lo que significa la característica hija debe hacer parte del producto (Figura 10 (c)). La cardinalidad equivalente a la relación opcional es [0..1], lo que significa que la característica hija, puede o no, hacer parte del producto (Figura 10 (d)).

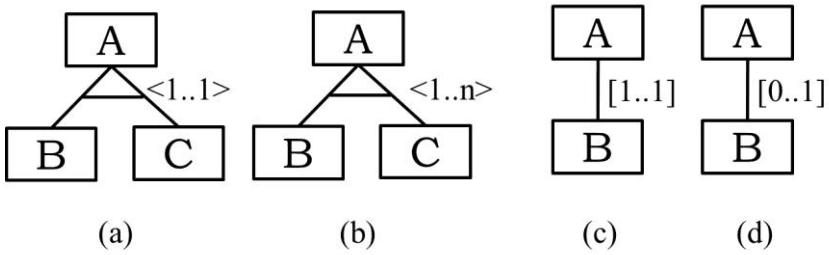


Figura 10 Relación de cardinalidad y multiplicidad.

- Relación no jerárquica.
  - Excluye: Una característica A excluye B significa que si la característica A es incluida en el producto, la característica B no debe ser incluida, y viceversa (Figura 11).
  - Requiere: Una característica A requiere B significa que si la característica A es incluida en el producto, la característica B también debe ser incluida, pero no viceversa (Figura 11).

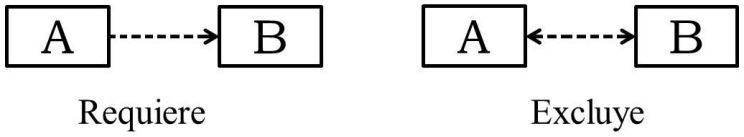


Figura 11 Relación no jerárquica entre características.

2.3. Procedimiento para la obtención del modelo variabilidad en LPS

Un procedimiento es un conjunto de acciones u operaciones que tienen que realizarse de la misma forma, para obtener siempre el mismo resultado bajo las mismas circunstancias. A continuación se describen brevemente las actividades necesarias para obtener el modelo de variabilidad.

- Identificar los procesos principales de la LPS. Esta actividad captura los procesos principales de la familia de productos a partir de toda la documentación e información recopilada de la LPS.
- Obtener aspectos comunes y aspectos variables. Su principal tarea es definir los aspectos comunes a todos los productos de la LPS y los aspectos variables que permiten la derivación de los aspectos personalizados para los diferentes productos.
- Determinar componentes comunes, variantes y opcionales. Se enmarca en capturar los aspectos comunes en los componentes comunes y los aspectos variables en los componentes variantes y opcionales dependiendo de la táctica que posee cada componente de la LPS.
- Obtener el comportamiento del componente. Puesto que parte del comportamiento del componente es común y parte de su comportamiento cambia dependiendo del producto, se precisa el comportamiento de cada componente, es decir se detalla la información técnica del componente, dando paso a su variabilidad interna.
- Determinar la variabilidad interna. Su principal tarea es especificar la variabilidad dentro de los componentes, a partir de la descripción basada en el objetivo y principales características del componente relacionado de forma directa con la funcionalidad que apoya el componente.
- Especificar las características y sus relaciones. Describe las características de todos los productos de la LPS y relaciona la característica padre con las características hijas que forman parte de cada producto.
- Representar el árbol jerárquico. Esta actividad tiene la responsabilidad de representar las características y las relaciones entre ellas a partir de una estructura jerárquica, es decir, mediante un árbol jerárquico. Este árbol se compone por una característica raíz que de ella cuelgan el resto de las características.
- Obtener el modelo de variabilidad. Define la variabilidad de la LPS.

Este procedimiento se representa como un proceso de negocio, siguiendo la notación de un diagrama de proceso de negocio (Uso de artefactos, actividades, eventos, compuertas y swimlanes; para la conexión gráfica se utilizó la secuencia de flujos normal). Un proceso de negocio es un conjunto de tareas relacionadas lógicamente, llevadas a cabo para generar

productos y servicios. La (Figura 12) muestra el procedimiento del modelo de variabilidad a partir de un diagrama de proceso de negocio.

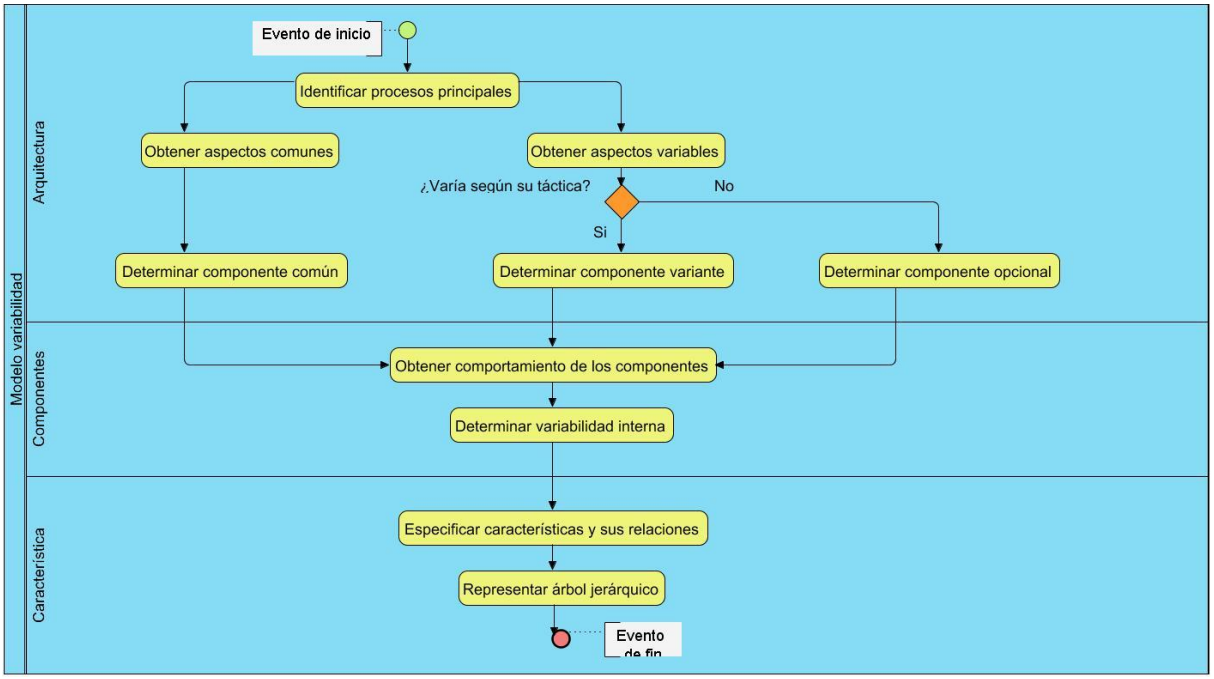


Figura 12 Procedimiento del modelo de variabilidad.

2.4. Modelo de variabilidad de Xedro-ERP

Xedro-ERP está integrado por 7 productos. A continuación se describe el modelo de variabilidad para Contabilidad, Capital Humano y Finanzas siguiendo el procedimiento explicado en la sección anterior.

Esta perspectiva describe la variabilidad posible de Xedro-ERP a nivel de arquitectura, componentes y características de los productos seleccionados de la línea. (Figura 13) (Figura 14)

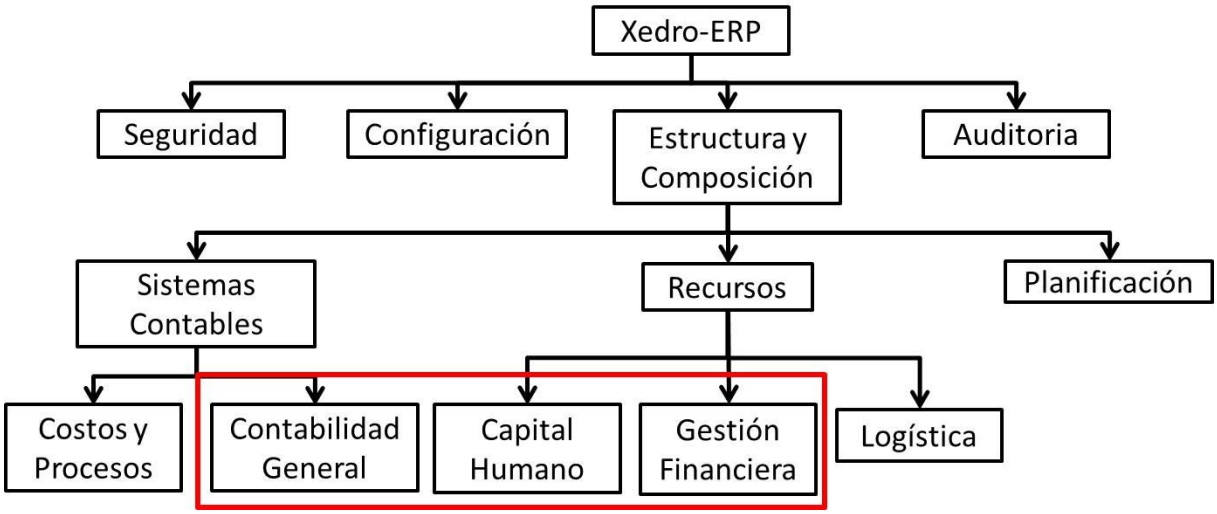


Figura 13 Vista general de Xedro-ERP.

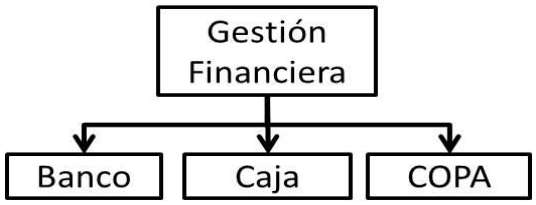


Figura 14 Vista del producto Gestión Financiera.

2.4.1. Variabilidad en arquitectura

La arquitectura de la línea de productos Xedro-ERP, está organizada en 3 vistas las cuales a su vez pueden descomponerse en otras vistas. La (Figura 15) muestra la estructura de la arquitectura de los productos de Contabilidad, Capital Humano y Finanzas que corresponden a la línea Xedro-ERP. En esta investigación llamaremos componentes a las vistas de la arquitectura definida por los productos seleccionados.

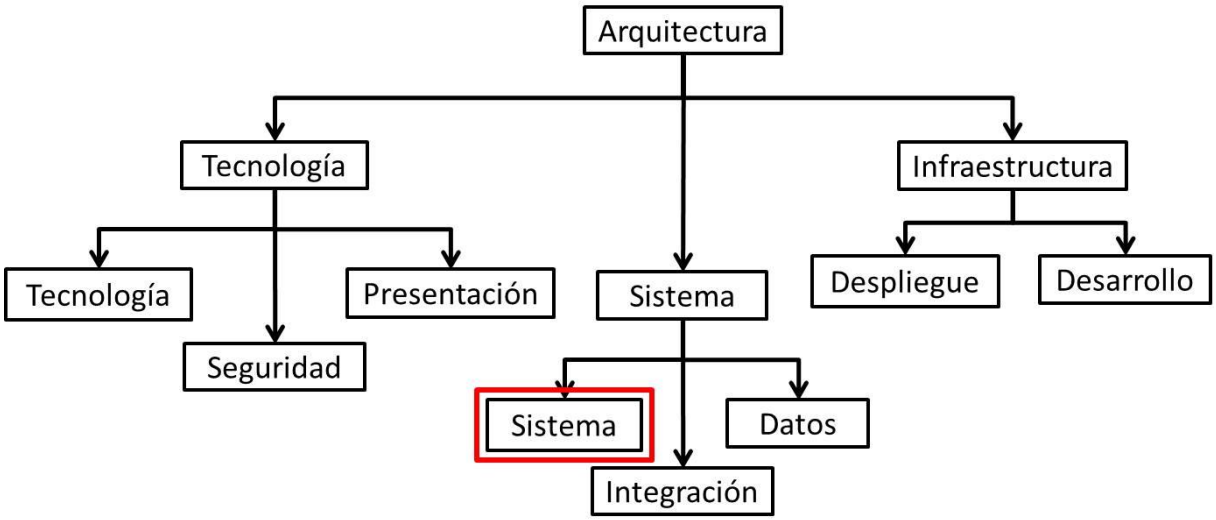


Figura 15 Arquitectura de Contabilidad, Capital Humano y Finanzas.

En esta investigación se dividirá la arquitectura en componentes comunes, opcionales y variantes. El componente Sistema obtiene los componentes que conforman Contabilidad, Capital Humano y Finanzas. Por esta razón, nuestro trabajo se enfocará en el componente Sistema.

Se afrontará los tipos de componentes que conforman cada producto seleccionado de Xedro-ERP y como estos se relacionan, así como de la composición estructural interna de cada uno de estos componentes, aunque de esto último estaremos abordando en el siguiente epígrafe que aborda de la variabilidad a nivel de componentes, es decir, la especificación interna de los componentes existentes en los productos de Contabilidad, Capital Humano y Finanzas.

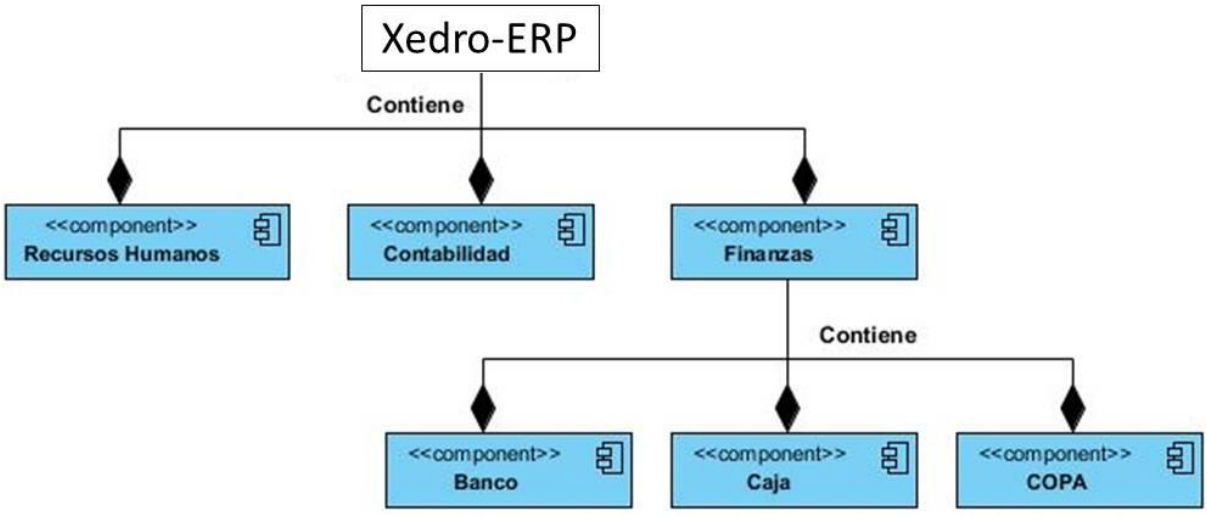


Figura 16 Mapa general de componentes Xedro-ERP.

Como se puede observar en la (Figura 16) estos componentes representan componentes opcionales. De los cuales se incorporan otros componentes definidos dentro de cada uno de estos componentes. Los componentes comunes para los componentes de Contabilidad, Recursos humanos y Finanzas (Banco, Caja, COPA) son los siguientes:

- Seguridad. Este componente chequea y logra una administración segura y centralizada de todos los aspectos relacionados con el acceso a la aplicación. Permite la seguridad a nivel de:
  - Autenticación
  - Autorización
  - Auditoría
  - Administración de perfiles
  - Administración de conexiones
- Tecnología. En este componente se define la plataforma tecnológica sobre la que se desarrolla estos componentes, donde se especifica lenguaje de programación, servidor web, entre otras tecnologías necesarias para su desarrollo. Estas son:
  - Apache como servidor web.
  - PHP como lenguaje de programación.
  - PostgreSQL como Gestor de Base de Datos.

- Mozilla Firefox como explorador web.
- Visual Paradigm for UML para el modelado.
- EXTJS para interfaz de usuario.
- Subversión para manejar el control de versiones.
- Doctrine

- Las interfaces utilizan para la conexión entre componentes IoC. Los IoC representan componentes en el que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

Contabilidad, Recursos humanos y Finanzas (Banco, Caja, COPA) también contienen dentro de ellos componentes variantes y opcionales. A continuación se describirá cada componente.

La imagen de la (Figura 17) describe el mapa general de componentes de Banco. Modelo que contendrá todos los componentes definidos en Banco, así como las distintas dependencias existentes entre los componentes.

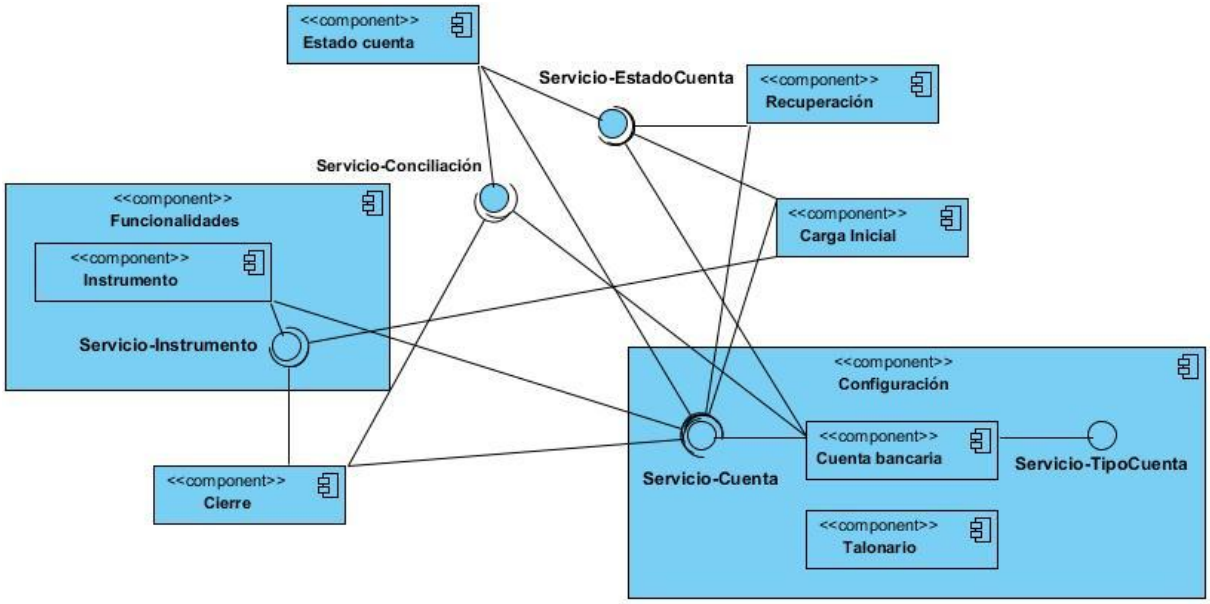


Figura 17 Mapa general de componentes Banco.

Como se observa en la (Figura 17) Banco está compuesto por 9 componentes relacionados mediante 5 interfaces proveídas y requeridas. Los componentes Recuperación, Cierre, Configuración, Carga inicial, Funcionalidades representan componentes variantes de los

cuales los miembros de la familia de Xedro-ERP emplean comportamientos diferentes, excepto el componente Carga inicial y Funcionalidades que son componentes variantes para Finanza. Los componentes Estado cuenta, Cuenta bancaria, Talonario y el componente Instrumento representan componentes opcionales los cuales son requeridos por Banco un componente de Finanzas, miembro de la familia de Xedro-ERP.

La imagen de la (Figura 18) describe el mapa general de componentes de Caja. Modelo que contendrá todos los componentes definidos en Caja, así como las distintas dependencias existentes entre los componentes.

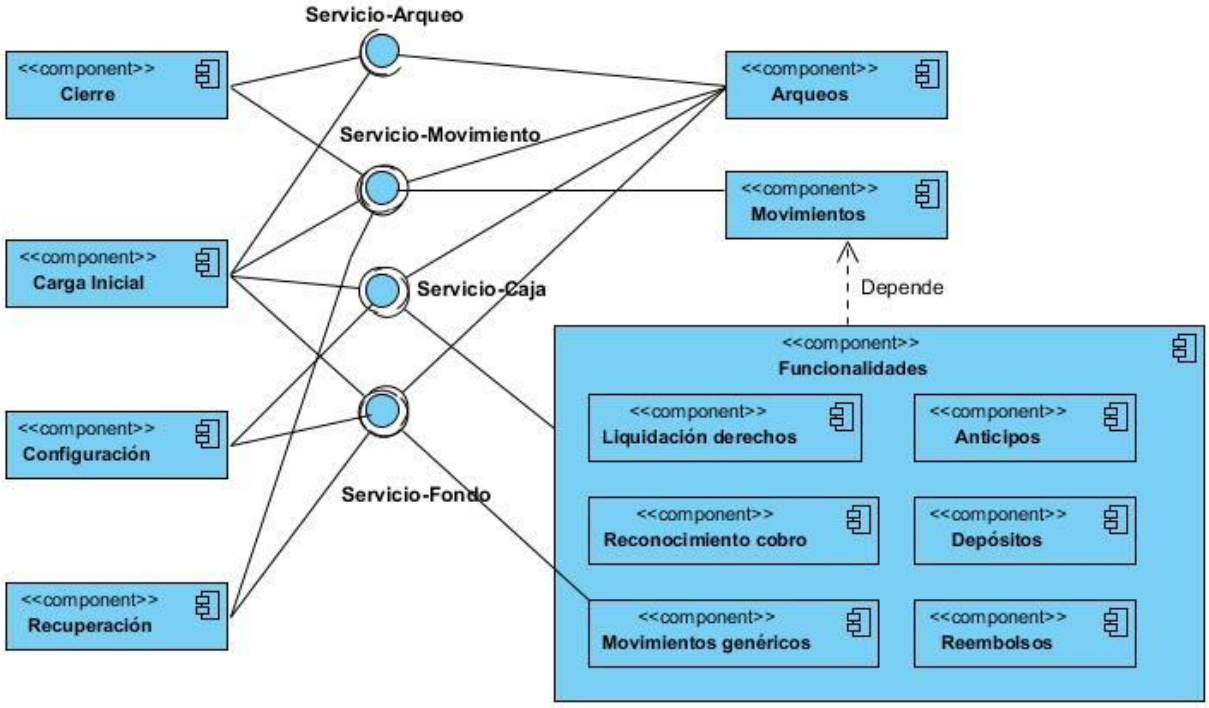


Figura 18 Mapa general de componentes Caja.

Como se observa en la (Figura 18) Caja está compuesto por 13 componentes relacionados mediante 4 interfaces proveídas y requeridas. Los componentes Recuperación, Cierre, Configuración, Funcionalidades y Carga inicial representan componentes variantes de los cuales los miembros de la familia de Xedro-ERP emplean comportamientos diferentes, excepto el componente Carga inicial y Funcionalidades que son componentes variantes para Finanza. Los componentes Arqueos, Movimientos, Liquidación derecho, Reconocimiento cobro, Reembolso, Depósito, Anticipos y Movimientos genéricos representan componentes opcionales los cuales son requeridos por Caja un componente de Finanzas, miembro de la familia de Xedro-ERP.

La imagen de la (Figura 19) describe el mapa general de componentes de COPA. Modelo que contendrá todos los componentes definidos en COPA, así como las distintas dependencias existentes entre los componentes.

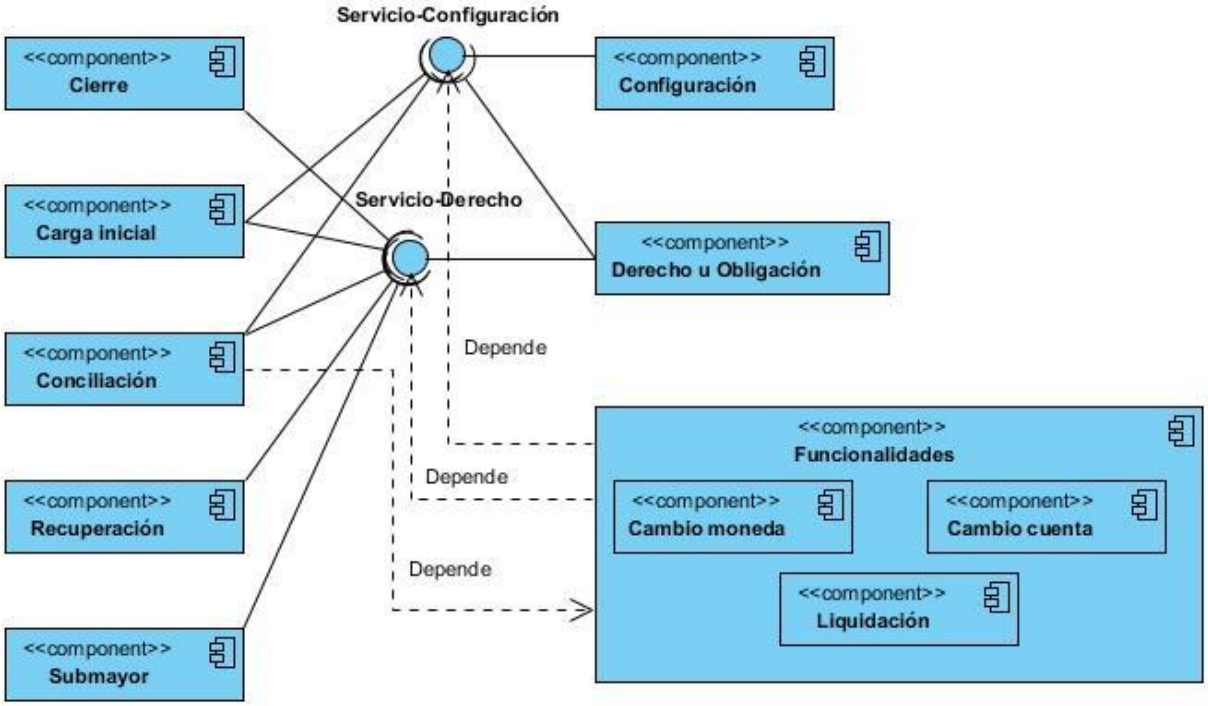


Figura 19 Mapa general de componentes COPA.

Como se observa en la (Figura 19) COPA está compuesto por 11 componentes relacionados mediante 2 interfaces proveídas y requeridas y a la vez por dependencias entre ellos. Los componentes Recuperación, Cierre, Configuración, Funcionalidades y Carga inicial representan componentes variantes de los cuales los miembros de la familia de Xedro-ERP emplean comportamientos diferentes, excepto el componente Carga inicial y Funcionalidades que son componentes variantes para Finanza. Los componentes Conciliación, Submayor, Derecho u obligación, Cambio moneda, Liquidación, Cambio cuenta representan componentes opcionales los cuales son requeridos por COPA un componente de Finanzas, miembro de la familia de Xedro-ERP.

La imagen de la (Figura 20) describe el mapa general de componentes de Capital Humano. Modelo que contendrá todos los componentes definidos en Capital Humano, así como las distintas dependencias existentes entre los componentes.

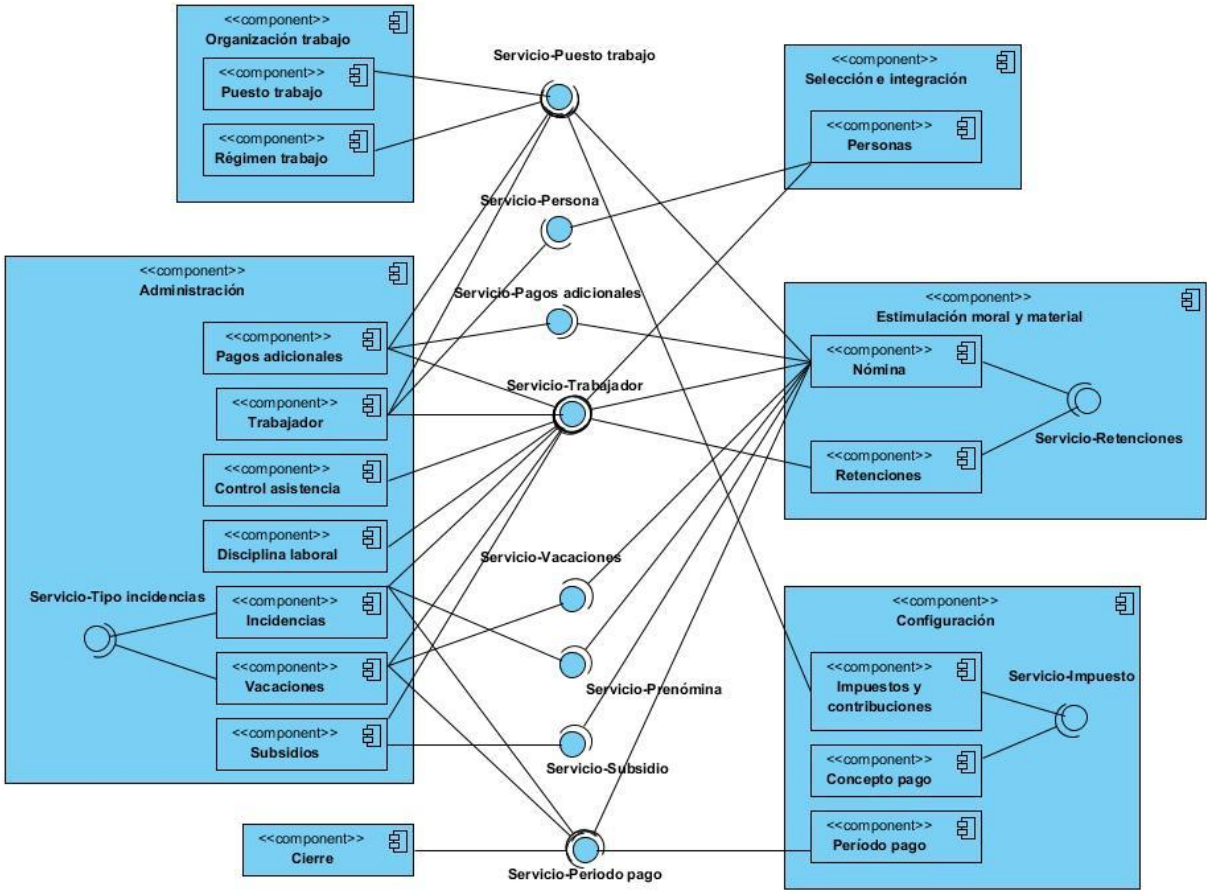


Figura 20 Mapa general de componentes Capital Humano.

Como se observa en la (Figura 20) Capital Humano está compuesto por 21 componentes relacionados mediante 8 interfaces proveídas y requeridas. Los componentes Configuración y Cierre representan componentes variantes de los cuales los miembros de la familia de Xedro-ERP emplean comportamientos diferentes. Los componentes Pagos adicionales, Trabajador, Control asistencia, Disciplina laboral, Incidencias, Vacaciones, Subsidios, Impuestos contribuciones, Concepto pago, Período pago, Nómina, Retenciones, Personas, Puesto trabajo y Régimen trabajo representan componentes opcionales los cuales son requeridos por Capital Humano un miembro de la familia de Xedro-ERP.

La imagen de la (Figura 21) describe el mapa general de componentes de Contabilidad. Modelo que contendrá todos los componentes definidos en Contabilidad, así como las distintas dependencias existentes entre los componentes.

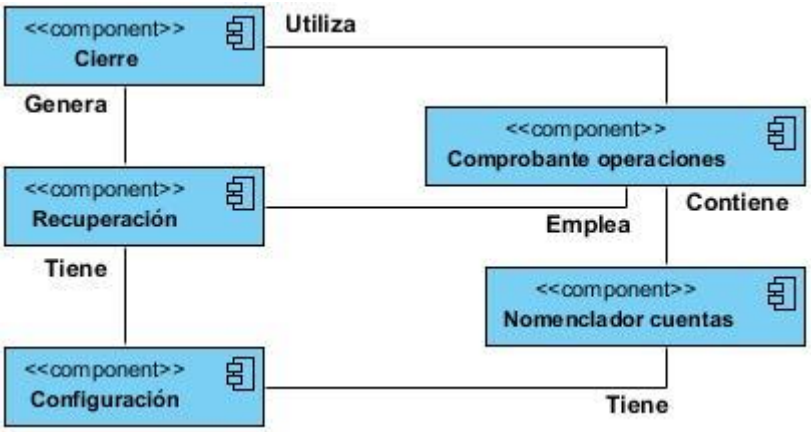


Figura 21 Mapa general de componentes Contabilidad. Tiene

Como se observa en la (Figura 21) Contabilidad está compuesto por 5 componentes relacionados entre ellos a partir de relaciones de generación, uso, entre otras. Los componentes Recuperación, Cierre y Configuración representan componentes variantes de los cuales los miembros de la familia de Xedro-ERP emplean distintas versiones. Los componentes Comprobante operaciones y Nomenclador cuentas representan componentes opcionales los cuales son requeridos por Contabilidad un miembro de la familia de Xedro-ERP.

2.4.2. Variabilidad en componentes

En esta sección se listan todos los componentes definidos por Contabilidad, Capital Humano Finanzas, y de cada uno de ellos se especificará una descripción basada en el objetivo y principales características del componente relacionado de forma directa con la funcionalidad que apoya el componente; los servicios que provee ese componente con sus detalles, es decir, muestra de forma más detallada la información técnica del componente. Además se especificará aspectos como flujos de información que el componente transfiere, usa y transforma. A continuación se listan los componentes con su descripción:

Componentes variantes

Recuperación

En el caso de Contabilidad el componente brinda servicios como: GetPases, que este se encarga de devolver los pases; GetAsientos, su función es devolver los asientos; VIMPComprobante, este último permite imprimir datos del comprobante. Este componente

es el encargado de gestionar todos los reportes del subsistema, dando la información necesaria y oportuna para la toma de decisiones.

En el caso de Banco, Caja y COPA que corresponden al componente Finanzas, el componente Recuperación es el encargado de gestionar los reportes del producto.

El componente Recuperación tiene 43 requisitos donde Banco está compuesto por 8 requisitos, Caja por 13 requisitos, COPA por 15 requisitos y Contabilidad por 7 requisitos.

Cierre

En el caso de Banco el componente se encarga de la realización del cierre diario, del cierre del período contable y del ejercicio fiscal. En Caja este componente es responsable por las funcionalidades de cierre de período de caja, cierre de período de subsistema, cierre de ejercicio de subsistema. Mientras que para COPA, el componente Cierre contiene la funcionalidad realizar cierre de período contable y de ejercicio fiscal del subsistema.

En el caso de Capital Humano es el encargado de realizar el cierre de los periodos de pago, contables y ejercicios contables. Mientras que para Contabilidad, el componente Cierre es el encargado de cerrar los Períodos y Ejercicios contables, y a la vez darle apertura a los próximos.

El componente Cierre tiene 18 requisitos donde Banco está compuesto por 6 requisitos, Caja por 3 requisitos, COPA por 1 requisito, Capital Humano por 3 requisitos y Contabilidad está compuesta por 5 requisitos.

Funcionalidades

En el caso de Banco, Caja y COPA que corresponden al componente Finanzas, el componente Funcionalidades se comporta de la siguiente manera:

Para Banco el componente brinda 22 servicios. Este componente es el encargado de gestionar todos los instrumentos bancarios. Debido a esto el componente Instrumento se encuentra dentro de este componente. En Caja el componente tiene la responsabilidad de realizar los: anticipos, reembolsos, movimientos genéricos y depósitos. Debido a esto los componentes Depósito, Anticipo, Reembolso, Liquidación derechos, Reconocimiento cobros y Movimientos genéricos se encuentran dentro de este componente. Mientras que para COPA tiene la responsabilidad de realizar los: cabios de moneda, cambios de cuenta de derechos y obligaciones, cancelaciones de contrato y derechos u obligaciones, así como sus

respectivas contabilizaciones. Debido a esto los componentes Cambio cuenta, Cambio moneda y Liquidación derecho se encuentran dentro de este componente. Además de brindar 2 servicios.

Configuración

En el caso de Banco el componente brinda 18 servicios. Este componente es el encargado de gestionar las cuentas, tipos de cuentas y talonarios. Debido a esto los componentes Cuenta bancaria y Talonario se encuentran dentro de este componente.

En el caso de Caja el componente brinda 33 servicios entre ellos se encuentran: NomencladoresService(DameCajas), el cual devuelve las cajas que pertenecen a una entidad; NomencladoresService(cargarfondos), este devuelve los fondos que administra una caja; NomencladoresService(realizarmovimiento), este último guarda la información referente a una movimiento de caja realizado; entre otros. En este componente se gestionan una serie de nomencladores.

En el caso de COPA el componente brinda 9 servicios entre ellos se encuentran: ConfiguracionService(Motivo), servicio para obtener los motivos de cancelación; ConfiguracionService(cargarParrafoFiscal), este servicio devuelve un párrafo fiscal. En este componente se configuran las condiciones de pago por defecto que utiliza la entidad y los motivos por los cuales se pueden realizar cancelaciones de documentos.

En el caso de Capital Humano el componente brinda 8 servicios. Este componente es el encargado de gestionar los impuestos y contribuciones, los períodos de pago y conceptos de pago. Debido a esto los componentes Impuestos y contribuciones, Períodos pago y Concepto pago se encuentran dentro de este componente.

En el caso de Contabilidad el componente brinda 16 servicios entre ellos se encuentran: ObtenerContenidosEconomicos, este servicio retorna el contenido económico dado el id; ObtenerNaturaleza, este retorna la naturaleza dado un id; ObtenerEstructuraE, este último retorna la estructura económica dado un id. En este componente se gestionan una serie de nomencladores que son la base para el componente Nomenclador de cuentas, entiéndase Grupos y subgrupos contables, Contenidos económicos de las cuentas y Estructuras económicas que representan a los propietarios de esas cuentas.

El componente Configuración tiene 70 requisitos donde Banco está compuesto por 19 requisitos, Caja por 5 requisitos, COPA por 5 requisitos, Capital Humano por 9 requisitos y Contabilidad está compuesta por 32 requisitos.

Carga inicial

En el caso de Banco, Caja y COPA que corresponden al componente Finanzas, el componente Carga inicial se comporta de la siguiente manera:

Para Banco el componente ingresa al sistema la información acumulada antes del momento de iniciar el procesamiento en el sistema. En Caja este componente chequea y carga los resultados de las operaciones antes del momento de inicio de procesamiento en el sistema, movimientos genéricos, anticipo de dieta para viaje nacional, anticipo de dieta para viaje al exterior, arqueo y cierre. Mientras que para COPA, el componente Carga inicial tiene la responsabilidad de introducir al sistema todos los derechos y obligaciones pendientes a liquidar total o parcialmente una vez que se instala el sistema.

El componente Carga inicial tiene 35 requisitos donde Banco está compuesto por 5 requisitos, Caja por 8 requisitos y COPA está compuesta por 22 requisitos.

Componentes opcionales

Comprobante operaciones

Este componente corresponde a Contabilidad y es el encargado de contabilizar los Comprobantes de operaciones de todos los subsistemas que componen la solución contable en desarrollo. Brinda 17 servicios entre ellos podemos citar a: CuentaSaldoPeriodo, este es el encargado de retornar el saldo de una cuenta en un período de un ejercicio dado; Vcomprobantet, servicio que muestra la interfaz del comprobante tipo de operaciones; cargarPaseCuenta, este retorna la cantidad de pases de una cuenta. El componente contiene 26 requisitos donde cada uno posee una funcionalidad diferente.

Nomenclador cuentas

Este componente corresponde a Contabilidad y es el encargado de gestionar el árbol de cuentas en el mayor sentido de la palabra, entiéndase las cuentas y sus aperturas, sean del tipo que sean, controladas por otros subsistemas. Brinda 12 servicios entre ellos podemos citar a: ObtenerCuentasNominales, este servicio retorna las cuentas nominales de una estructura dada; ObtenerContenidoCuenta, este retorna el contenido económico de una

cuenta dada; CargarCuentaTipo, este último retorna las cuentas tipo dado una estructura. El componente contiene 22 requisitos donde cada uno posee una funcionalidad diferente.

Concepto pago

Este componente está comprendido dentro del componente Configuración y corresponde a Capital Humano y gestiona los conceptos de pago. Brinda 3 servicios ellos son: chequearConceptos, comprueba que el concepto de pago tiene el id de la cuenta, el id de la operación y el id del elemento de gasto asociado; obtenerConcepto, servicio que obtiene datos de concepto de pago y BuscarConceptoDeImpuestos, servicio que comprueba si algún concepto de pago tiene asociado el impuesto indicado. El componente contiene 2 requisitos donde cada uno posee una funcionalidad diferente.

Período pago

Este componente está comprendido dentro del componente Configuración y corresponde a Capital Humano y gestionar los períodos de pago. Brinda 5 servicios entre ellos podemos citar a: obtenerPeriodosAbiertos, servicio que obtiene los períodos de pago abiertos; PeriodosDePago, devuelve todos los períodos de pago abiertos y CerrarPeriodoPago, cierra el período de pago actual y pone como actual el siguiente abierto. El componente contiene 7 requisitos donde cada uno posee una funcionalidad diferente.

Nómina

Este componente está comprendido dentro del componente Estimulación moral y material y corresponde a Capital Humano y gestiona todo lo referente a las nóminas. Brinda 3 servicios ellos son: obtenerDenominacionNominas, servicio que obtiene las denominaciones de las nóminas dado un arreglo de ids; buscarRetencionenUso, servicio que sabe si un registro de retención está en uso en el componente nómina y NominasTodasContabilizadas, devuelve la cantidad de nóminas del período que faltan por contabilizar para esa estructura. El componente contiene 29 requisitos donde cada uno posee una funcionalidad diferente.

Retenciones

Este componente está comprendido dentro del componente Estimulación moral y material y corresponde a Capital Humano y gestiona todo lo referente a las retenciones. Brinda 2 servicios ellos son: RentencionesDelTrabajador, devuelve todas las retenciones activas del trabajador y ActualizarSubmayor, servicio que actualiza el submayor con el registro de los

descuentos realizados. El componente contiene 14 requisitos donde cada uno posee una funcionalidad diferente.

Subsidio

Este componente está comprendido dentro del componente Administración y corresponde a Capital Humano y gestiona los subsidios. Brinda 2 servicios ellos son: SubsidiosDeTrabporPeriodo, devuelve las notificaciones de subsidio en estado confirmado dado un período de pago y CambiarEstadoNotificaciones, se cambia a un estado un grupo de notificaciones de subsidio. El componente contiene 33 requisitos donde cada uno posee una funcionalidad diferente.

Vacaciones

Este componente está comprendido dentro del componente Administración y corresponde a Capital Humano y gestiona las vacaciones. Brinda 6 servicios entre ellos podemos citar a: insertarOperaciones, servicio que insertar una operación en el submayor de vacaciones (actualizar submayor); getPlanDeVacacionesConfirmados, se obtienen las notificaciones de vacaciones confirmadas en el período de pago actual en la entidad y eliminarPlanPorIdTrab, servicio que elimina un plan de vacaciones dado el id del trabajador. El componente contiene 22 requisitos donde cada uno posee una funcionalidad diferente.

Incidencias

Este componente está comprendido dentro del componente Administración y corresponde a Capital Humano y gestiona todo lo referente a las incidencias incluyendo las pre-nóminas. Brinda 15 servicios entre ellos podemos citar a: ObtenerTipoIncidencia, devuelve un tipo de incidencia dado el id o dado el parámetro en específico, sino las devuelve todas; InsertarIncidencia, permite insertar un nuevo tipo de incidencia y ObtenerPrenomina, devuelve todos los datos de todas las pre-nóminas. El componente contiene 19 requisitos donde cada uno posee una funcionalidad diferente.

Disciplina laboral

Este componente está comprendido dentro del componente Administración y corresponde a Capital Humano y gestiona todo lo referente a la disciplina laboral. Brinda 1 servicio, el cual es: ObtenerPenalizacionesTrabajador, servicio que devuelve los registros de medidas disciplinarias que tenga el trabajador. El componente contiene 18 requisitos donde cada uno posee una funcionalidad diferente.

Control asistencia

Este componente está comprendido dentro del componente Administración y corresponde a Capital Humano y gestiona todo lo referente al control de asistencia. El componente contiene 10 requisitos donde cada uno posee una funcionalidad diferente.

Conciliación

Este componente corresponde a COPA de Finanzas. Este componente tiene la responsabilidad de realizar las conciliaciones entre entidades, de derechos y obligaciones, así como operaciones anticipadas, pudiendo llegar a cancelaciones por expediente y cambios de plazos en caso que sea conveniente. El componente contiene 4 requisitos donde cada uno posee una funcionalidad diferente.

Submayor

Este componente corresponde a COPA de Finanzas. Es el encargado de mantener actualizados los submayores fiscales y de clientes y proveedores. Brinda 2 servicios ellos son: SubmayorService(InsertarCierre), este servicio al cerrar un período se inserta en la tabla el saldo con el que queda cada cuenta en ese período y el servicio SubmayorService(SaldoTotalXCuentasPeriodo), que dado un listado de cuentas devuelve el saldo que tiene cada cuenta en ese período en Finanzas. El componente contiene 6 requisitos donde cada uno posee una funcionalidad diferente.

Derecho u obligación

Este componente corresponde a COPA de Finanzas. Este componente tiene la responsabilidad de registrar, modificar, cancelar, confirmar y contabilizar los derechos y obligaciones que la entidad contrae con terceros. Brinda 77 servicios entre ellos podemos citar a: DerechoService(CargarOperacion), este servicio devuelve la información referente a una operación; DerechoService(ObtenerDerecho), servicio que devuelve los datos referentes a un derecho y DerechoService(Idclientes), busca los clientes que han efectuado algún derecho u obligación. El componente contiene 22 requisitos donde cada uno posee una funcionalidad diferente.

Cambio moneda

Este componente está comprendido dentro del componente Funcionalidades y corresponde a COPA de Finanzas. Tiene la responsabilidad de realizar los cabios de moneda. El componente contiene 4 requisitos donde cada uno posee una funcionalidad diferente.

Liquidación

Este componente está comprendido dentro del componente Funcionalidades y corresponde a COPA de Finanzas. Tiene la responsabilidad de realizar las liquidaciones de los pagos cobros anticipados, así como sus respectivas contabilizaciones. El componente contiene 2 requisitos donde cada uno posee una funcionalidad diferente.

Cambio cuenta

Este componente está comprendido dentro del componente Funcionalidades y corresponde a COPA de Finanzas. Tiene la responsabilidad de realizar los cambios de cuenta de derechos y obligaciones. Brinda 2 servicios ellos son: DerechoService(ContabilizarCambiodeCuentaDO), servicio para derecho u obligación y DerechoService(ContabilizarCambiodeCuentaCP), servicio para cobro o pago anticipado. El componente contiene 4 requisitos donde cada uno posee una funcionalidad diferente.

Arqueos

Este componente corresponde a Caja de Finanzas. Este componente realiza las funciones de arqueo de efectivo en caja por depositar, y arqueo de efectivo en caja. Brinda 16 servicios entre ellos podemos citar a: ModificarArqueo, servicio con el que se modifican los arqueos realizados; EliminarArqueoCI, servicio con el que se eliminan los arqueos realizados en la Carga Inicial y ArqueoService(CargarArqueos), servicio con el que se obtienen todos los arqueos elaborados. El componente contiene 6 requisitos donde cada uno posee una funcionalidad diferente.

Movimientos

Este componente corresponde a Caja de Finanzas. Este componente soporta las operaciones del paquete Funcionalidades, gestionar anticipo de dieta para viaje nacional, anticipo de dieta para viaje al exterior, liquidar anticipo de dieta para viaje nacional, liquidar anticipo de dieta para viaje al exterior, y realiza las funcionalidades de depósito, reembolso y los movimientos genéricos. Brinda 86 servicios entre ellos podemos citar a:

MovimientoService(DameClasificacion), servicio para obtener la clasificación del anticipo; MovimientoService(DameAnticipos), servicio que devuelve los anticipos otorgados y MovimientoService(InsertarDeposito), servicio que guarda la información referente a una operación de depósito. El componente contiene 51 requisitos donde cada uno posee una funcionalidad diferente.

Liquidación derecho

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar liquidación de derechos. El componente contiene 2 requisitos donde cada uno posee una funcionalidad diferente.

Reconocimiento cobro

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar reconocimiento cobro. El componente contiene 1 requisitos donde cada uno posee una funcionalidad diferente.

Movimientos genéricos

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar movimientos genéricos. El componente contiene 11 requisitos donde cada uno posee una funcionalidad diferente.

Anticipos

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar anticipos. El componente contiene 17 requisitos donde cada uno posee una funcionalidad diferente.

Depósitos

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar depósitos. El componente contiene 6 requisitos donde cada uno posee una funcionalidad diferente.

Reembolsos

Este componente está comprendido dentro del paquete funcionalidades, que corresponde a Caja de Finanzas. Tiene la responsabilidad de realizar reembolsos. El componente contiene 11 requisitos donde cada uno posee una funcionalidad diferente.

Estado cuenta

Este componente corresponde a Banco de Finanzas. Gestiona los estados de las cuentas que llegan a la entidad desde el banco y es responsable del proceso de conciliación bancaria. Brinda 30 servicios entre ellos podemos citar a: EstadoCuentaService(ObtenerDiferencia), servicio que devuelve la información asociada a la diferencia correspondiente a una cuenta; EstadoCuentaService(cantidadEstadosCuenta), servicio que devuelve para la cuenta bancaria correspondiente al identificador recibido, la cantidad de estados por los que ha pasado y EstadoCuentaService(ObtenerEstadosTodos), servicio que devuelve todos los estados por los que puede transitar una cuenta bancaria. El componente contiene 26 requisitos donde cada uno posee una funcionalidad diferente.

Cuenta bancaria

Este componente está comprendido dentro del componente Configuración y corresponde a Banco de Finanzas. Se encuentra dentro del componente Configuración, y se encarga de la gestión de las cuentas bancarias, incluyendo el tipo de esta, que atiende la entidad.

Brinda 18 servicios entre ellos podemos citar a: CuentaService(DatosCuenta), servicio que obtiene todos los datos asociados a una cuenta bancaria; TipoCuentaService(ObtenerTiposCuenta), servicio que devuelve los tipos de cuentas de una cuenta bancaria y CuentaService(ObtenerDatosCuentas), servicio que devuelve la información referente a una cuenta bancaria. El componente contiene 14 requisitos donde cada uno posee una funcionalidad diferente.

Talonario

Este componente está comprendido dentro del componente Configuración y corresponde a Banco de Finanzas. Se encuentra dentro del componente Configuración, y se encarga de la gestión de los talonarios asignados a cada cuenta bancaria. El componente contiene 5 requisitos donde cada uno posee una funcionalidad diferente.

Instrumento

Este componente corresponde a Banco de Finanzas. Se ocupa de la gestión de los instrumentos correspondientes a una operación realizada, asociados a cada talonario perteneciente a una cuenta. Brinda 22 servicios entre ellos podemos citar a: InstrumentoService(ObtenerMovimientos), servicio que devuelve los movimientos realizados en una cuenta bancaria; InstrumentoService(Insertarmovimientobancario), servicio que

guarda los datos correspondientes a un movimiento bancario y InstrumentoService(operacionesporCuenta), servicio que devuelve las operaciones realizadas sobre una cuenta bancaria. El componente contiene 30 requisitos donde cada uno posee una funcionalidad diferente.

2.4.3. Variabilidad en características

El modelo de la (Figura 22) describe en forma de árbol las características de Banco donde la raíz representa el sistema como un todo. El árbol está formado por seis secciones principales (Estado cuenta, Carga inicial, Instrumento bancario, Configuración, Recuperaciones y Cierre) las cuales son obligatorias, excepto la sección de Recuperación que son opcional, lo que indica que puede o no estar presente en cualquier nuevo producto.

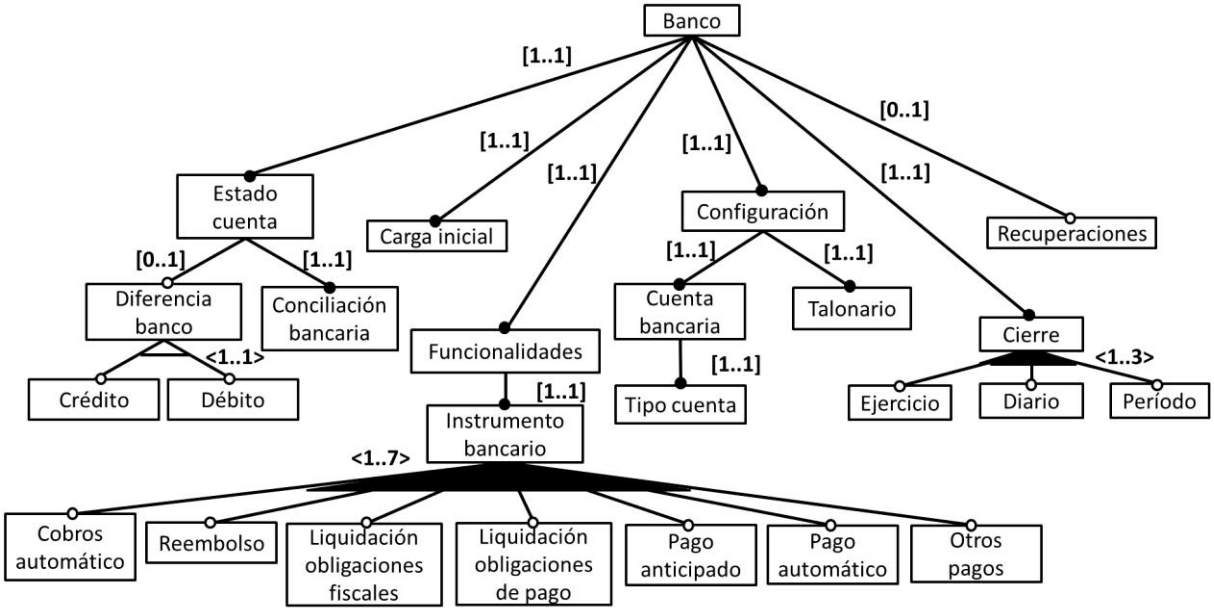


Figura 22 Variabilidad en características del componente Banco de Finanzas.

Como se puede observar en la (Figura 22) la relación Cierre con sus características hijas son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En la relación Funcionalidades- Instrumento bancario se puede ver una relación obligatoria lo cual indica, que Instrumento bancario debe estar presente de alguna forma pero como la relación Instrumento bancario con sus características hijas son de tipo OR esto indica que pueden estar presentes una, dos o todas las características a la vez. En la relación Estado cuenta-Diferencia banco con sus características hijas es alternativa esto indica que solo estará presente una de las dos diferencias con banco (Crédito o Débito). En

el árbol predomina la relación obligatoria lo cual indica, que deben de estar presentes de alguna forma.

El modelo de la (Figura 23) describe en forma de árbol las características de Caja donde la raíz representa el sistema como un todo. El árbol está formado por siete secciones principales (Arqueo, Configuración, Recuperación, Movimientos, Funcionalidades, Carga inicial y Cierre) las cuales son obligatorias excepto las secciones de Arqueo y Recuperación que son opcionales, lo que indica que puede o no estar presente en cualquier nuevo producto.

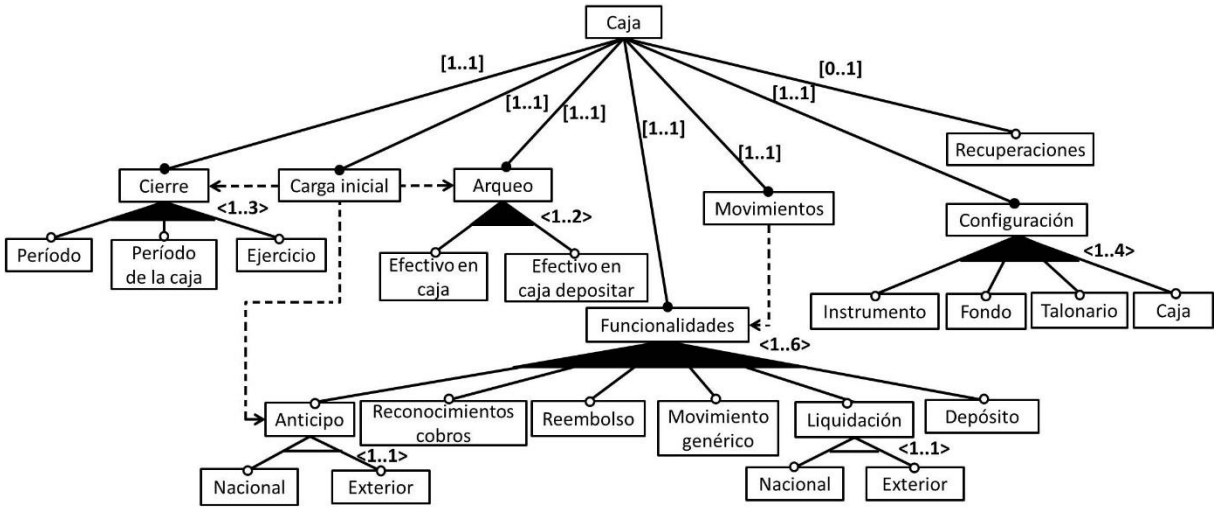


Figura 23 Variabilidad en características del componente Caja de Finanzas.

Como se puede observar en la (Figura 23) la relación Arqueo, Funcionalidades, Configuración y la relación Cierre con sus características hijas son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En las relaciones Funcionalidades-Anticipo y Funcionalidades -Liquidación con sus características hijas es alternativa esto indica que sólo estará presente una de los dos anticipos o liquidación (Nacional o Exterior). La relación Carga inicial requiere 3 relaciones que significa que si la característica Carga inicial es incluida en el producto, las características Cierre, Arqueo Anticipo también deben ser incluidas, pero no viceversa. También la relación Movimiento requiere Funcionalidades que significa que si la característica Movimiento es incluida en el producto, la característica Funcionalidades también debe ser incluida, pero no viceversa.

El modelo de la (Figura 24) describe en forma de árbol las características de COPA donde la raíz representa el sistema como un todo. El árbol está formado por ocho secciones

principales (Carga inicial, Configuración, Derecho u obligación, Conciliación, Recuperación, Funcionalidades, Submayor y Cierre) las cuales son obligatorias excepto la sección de Recuperación que es opcional, lo que indica que puede o no estar presente en cualquier nuevo producto.

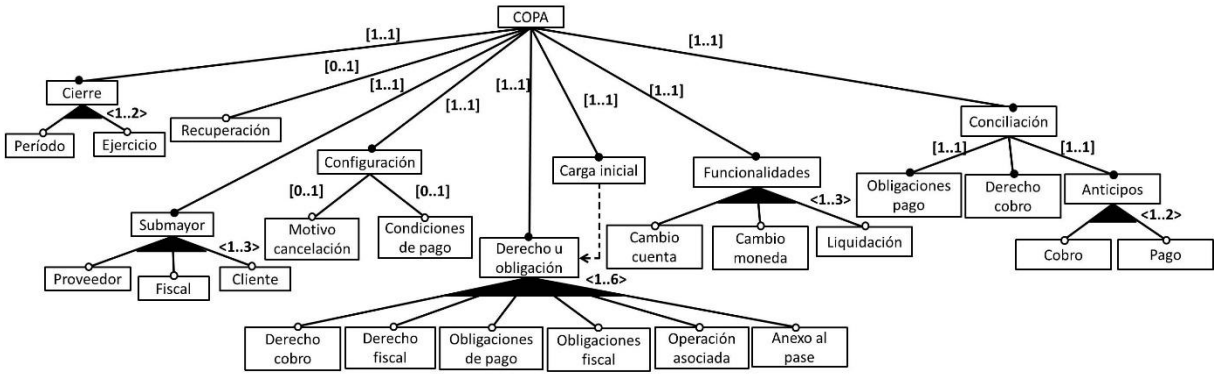


Figura 24 Variabilidad en características del componente COPA de Finanzas.

Como se puede observar en la (Figura 24) la relación Submayor, la relación Derecho u obligación, la relación Funcionalidades y la relación Cierre con sus características hijas son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En la relaciones Configuración se pueden ver dos relación opcionales las cuales indican, que pueden o no estar presente de alguna forma. La relación Conciliación-Derecho u obligación y la relación Conciliación-Anticipos se puede ver relaciones obligatorias lo cual indica, que Derecho u obligación Anticipos deben estar presentes de alguna forma. Aunque la relación Anticipos con sus características hijas son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. La relación Carga inicial requiere Derecho u obligación que significa que si la característica Carga inicial es incluida en el producto, la característica Derecho u obligación también debe ser incluida, pero no viceversa.

El modelo de la (Figura 25) describe en forma de árbol las características de Capital Humano donde la raíz representa el sistema como un todo. El árbol está formado por seis secciones principales (Organización trabajo, Selección e integración, Estimulación moral y material, Cierre, Administración y Configuración) las cuales son obligatorias, lo que indica que tienen que estar presente en cualquier nuevo producto.

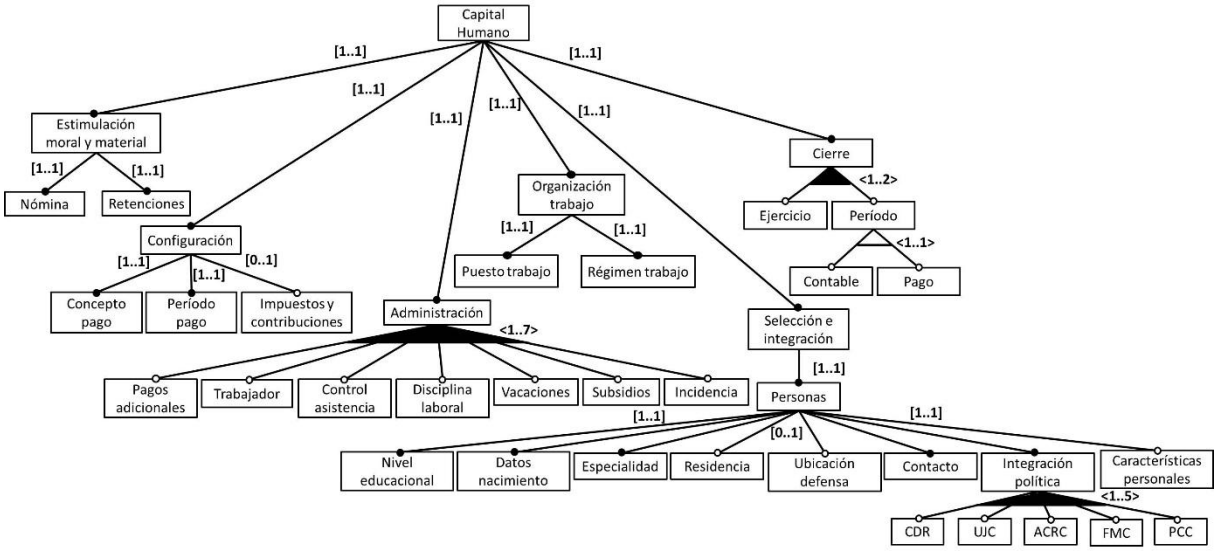


Figura 25 Variabilidad en características del componente Capital Humano.

Como se puede observar en la (Figura 25) la relación Administración y Cierre con sus características hijas son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En la relación Selección e integración-Personas-Integración política se puede ver una relación obligatoria lo cual indica, que Integración política debe estar presente de alguna forma pero como la relación Integración política con sus características son de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En las demás relaciones se puede ver relaciones obligatorias lo cual indica, que deben estar presentes de alguna forma excepto las relaciones Selección e integración-Personas-Residencia, Selección e integración-Personas-Ubicación defensa, Selección e integración-Personas-Características personales y la Configuración-Impuestos y contribuciones que son opcionales, lo que indica que puede o no estar presente en cualquier nuevo producto. En la relación Cierre-Período se puede ver una relación opcional lo cual indica, que Período debe o no estar presente pero como la relación Período con sus características hijas es alternativa esto indica que solo estará presente una de los dos períodos (Pago o Contable).

El modelo de la (Figura 26) describe en forma de árbol las características de Contabilidad donde la raíz representa el sistema como un todo. El árbol está formado por cinco secciones principales (Recuperación, Comprobante de operación, Nomenclador cuenta, Configuración y Cierre) las cuales son obligatorias excepto la sección de Recuperación que es opcional, lo que indica que puede o no estar presente en cualquier nuevo producto.

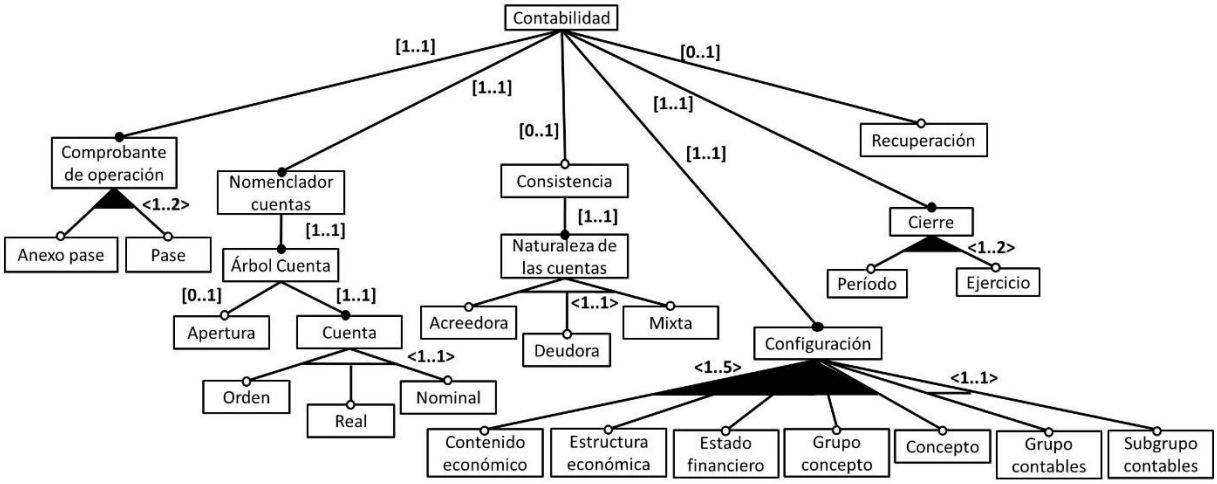


Figura 26 Variabilidad en características del componente Contabilidad.

Como se puede observar en la (Figura 26) la relación Configuración, Comprobante de operaciones y la relación Cierre con sus características hijas es de tipo OR que indica que pueden estar presentes una, dos o todas las características a la vez. En la relación Nomenclador cuentas-Árbol cuenta-Cuenta se puede ver una relación obligatoria lo cual indica, que Cuenta debe estar presente de alguna forma pero como la relación Cuenta con sus características hijas es alternativa esto indica que solo estará presente una de los tres cuentas (Orden, Real o Nominal). Además la relación Consistencia-Naturaleza de las cuentas se puede ver una relación obligatoria lo cual indica, que Naturaleza de las cuentas debe estar presente de alguna forma pero como la relación Naturaleza de las cuentas con sus características hijas es alternativa esto indica que solo estará presente una de los tres naturalezas (Acreedora, Deudora o Mixta). También en la relación Configuración está presente una relación alternativa lo cual indica que solo puede estar presente uno de los dos (Grupo o Subgrupo contables).

2.5. Resultados obtenidos

Luego de haber llevado a cabo el proceso de modelado en este epígrafe se hace indispensable documentar los resultados obtenidos. Con el modelado de la variabilidad de Xedro-ERP fue posible afirmar que los productos de la línea fueron desarrollados por separado. Por tal razón, a los productos como Contabilidad, Capital humano y Finanzas se le fueron implementando y desarrollando los componentes en la medida que se necesitaban específicamente. Esta implementación y desarrollo de los componentes fue alterando la

variabilidad de Xedro-ERP. Después del modelado quedo plasmado que la arquitectura de los productos analizados están comprendidas por componentes variantes y opcionales.

Los componentes comunes, como su nombre lo indica son aspectos comunes a todos los miembros de la familia de productos. Estos están relacionados con la Seguridad, donde cada producto analizado chequea todos los aspectos relacionados con el acceso a la aplicación; con la Tecnología utilizada en el desarrollo de Contabilidad, Capital humano y Finanzas. Otro aspecto común está relacionado con las interfaces pues estas utilizan para la conexión entre componentes IoC.

Debido a toda la variabilidad que existe entre estos productos, en la variabilidad a nivel de características se refleja detalladamente el comportamiento lógico de ellos. Por tal razón, cada producto pose características diferentes.

2.6. Conclusión

En el desarrollo de este capítulo se representó el modelo de variabilidad que propone analizar y guiar la personalización o integración de nuevos productos a la LPS. Lo explicado hasta el momento ofrece una amplia panorámica de la propuesta de solución. Sé aseguró de este modo el cumplimiento del objetivo primario de esta investigación al lograr representar la variabilidad de Xedro-ERP mediante un modelo de variabilidad.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

3.1. Introducción

En el presente capítulo se realizará la validación de la propuesta de solución ofrecida en el desarrollo del trabajo, teniendo en cuenta métricas donde se indicarán indicadores (preguntas) y se recogerán opiniones a los arquitectos y analistas de Xedro-ERP para dar un resultado negativo o positivo de la misma, determinando si está en condiciones de ser aplicada o no.

3.2. Validación de la solución

Lograr la aceptación y validación del modelo de variabilidad para garantizar el éxito de las nuevas personalizaciones realizadas a la línea Xedro-ERP del centro CEIGE consta de dos partes.

Como se mencionó anteriormente para reducir el tiempo de desarrollo de las nuevas personalizaciones realizadas a Xedro-ERP se precisa de la definición y aplicación de un modelo que permita el análisis de variabilidad de la línea. Donde las variables a validar son: tiempo de desarrollo de las nuevas personalizaciones (variable dependiente) y modelo de variabilidad (variable independiente).

3.2.1. Validación de la variable independiente

Métrica para validar el modelo de variabilidad.

Forma de Uso

- Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 0 en caso de mal (cuando la respuesta al indicador sea “No”) y 1 en caso que el elemento revisado no presente errores (cuando la respuesta al indicador sea “Sí”).
- Cantidad de elementos afectados (Cant_E): Especifica la cantidad de errores encontrados sobre el mismo indicador.
- Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Indicadores a evaluar	Eval	Cant_E	Comentarios
1. ¿La bibliografía consultada es actual?	1	_	
2. ¿Los conceptos están bien descritos?	1	_	
3. ¿Es una representación abstracta, gráfica, conceptual o visual?	1	_	
4. ¿Se detalla la forma de representación del modelo?	1	_	
5. ¿El modelo es factible de aplicar?	1	_	
6. ¿Se dan ejemplos en el modelo para facilitar su uso?	1	_	
7. ¿Se puede aplicar el modelo de variabilidad en otro contexto?	1	_	
8. ¿Un arquitecto usando el modelo puede determinar si un producto es válido para una LPS?	1	_	
9. ¿Un analista usando el modelo puede determinar si un producto es válido para una LPS?	1	_	
10. ¿Puede un arquitecto de la línea aplicar el modelo de variabilidad?	1	_	
11. ¿Puede un analista de la línea aplicar el modelo de variabilidad?	1	_	
12. ¿Se explica cómo aplicar el modelo?	1	_	

13. ¿Una vez aplicado el modelo facilita a los arquitectos al análisis de la LPS?	1	_	
14. ¿Una vez aplicado el modelo facilita a los analistas al análisis de la LPS?	1	_	
15. Se proponen 3 vistas fundamentales por las cuales transita el modelo para analizar la variabilidad de la línea. Categorice cada una de ellas:	1	_	
15.1. Variabilidad en arquitectura	1	_	
15.2. Variabilidad en componentes	1	_	
15.3. Variabilidad en características	1	_	
16. Contiene un procedimiento que explica a los arquitectos y analistas cómo aplicar el modelo.	1		
Total	No = 0	_	
	Si = 19		

Tabla 1 Métrica para evaluar el modelo de variabilidad.

Evaluación de la Métrica

- Se evalúa de **BIEN** si la cantidad de indicadores evaluados de bien es mayor de un 80% y la cantidad de elementos afectados de un indicador evaluado de mal no es mayor que 2.
- Se evalúa de **MAL** si la cantidad de indicadores evaluados de bien es menor de un 80% y la cantidad de elementos afectados de un indicador evaluado de mal es superior a tres.

3.2.2. Validación de la variable dependiente

Para la validación se realizó un cuestionario en el que se plantearon indicadores para recoger diferentes opiniones de especialistas de la línea de producto Xedro-ERP y potenciales usuarios o clientes de la propuesta presentada.

Los especialistas fueron seleccionados teniendo en cuenta las siguientes características:

- Graduado de nivel superior de informática.
- Tres años de experiencia como mínimo.
- Vinculación al desarrollo de productos informáticos.
- Conocimientos y habilidades en el funcionamiento de la línea de producto Xedro-ERP.
- Tres años desempeñando el rol de arquitecto o el rol de analista como mínimo.

Se estableció una evaluación en el cual cada especialista ofreció un nivel para cada indicador. La medición puede realizarse de varias formas: Si (S), No (N) o No sé (NS).

	S	N	NS
1. Los conceptos están descritos adecuadamente.			
2. Las descripciones de los conceptos son fáciles de entender.			
3. Se puede crear nuevas configuraciones con menor tiempo.			
4. Se puede definir los puntos reutilizables.			
5. Reúne todo la información de la línea.			
6. El tiempo de desarrollo de las nuevas personalizaciones se reduce.			

3.3. Resultados obtenidos

Esta sección se enfocó a los resultados obtenidos en la validación de la propuesta. El gráfico de la (Figura 27) refleja los resultados arrojados de la (Tabla1). Mientras que el gráfico de la (Figura 28) refleja los resultados de la validación del tiempo de desarrollo de las nuevas personalizaciones. A continuación se representan los resultados obtenidos:

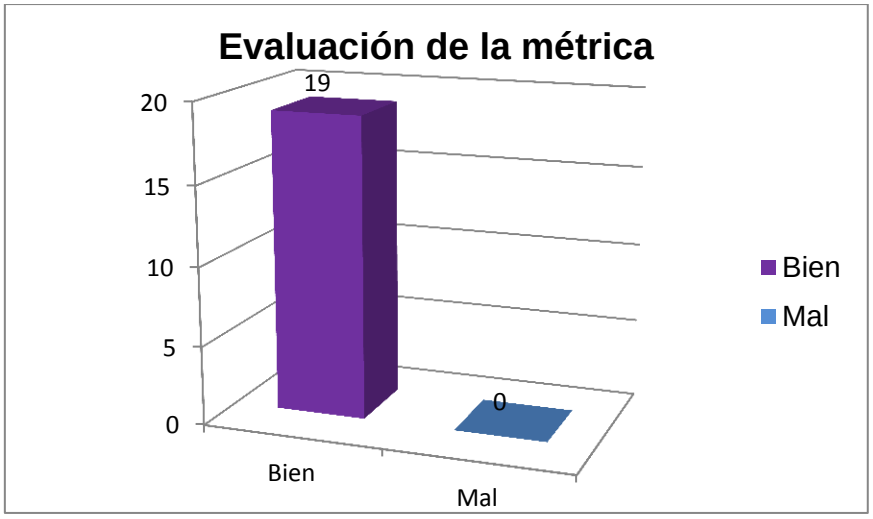


Figura 27 Resultados finales de la validación del modelo de variabilidad.

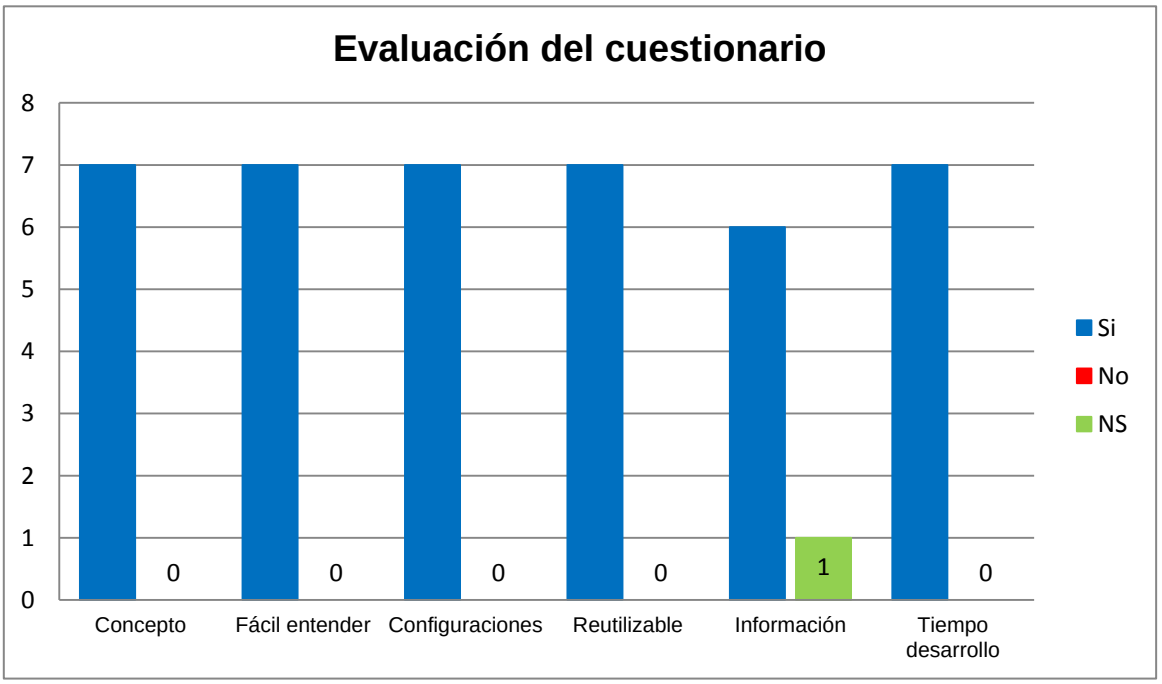


Figura 28 Resultados finales de la validación del tiempo de desarrollo.

Con la aplicación de la validación se detectó que el procedimiento sometido al modelo de variabilidad fue evaluado de Bien. Se recogieron opiniones de los especialistas de la línea de producto arrojando: que 7 profesionales opinaron que los conceptos están descritos adecuadamente y las descripciones de los conceptos son fáciles de entender, que se puede crear nuevas configuraciones con menor tiempo, se puede definir los puntos reutilizables y

que el tiempo de desarrollo de las nuevas personalizaciones se reduce al aplicar el modelo de variabilidad. Mientras que 6 profesionales opinaron que el modelo reúne toda la información de la línea. Aunque 1 profesional opinó que no sabe si el modelo reúne toda la información de la línea.

3.4. Premios otorgados a la solución

Como mecanismo para constatar la opinión de diversos especialistas en la temática y de la comunidad científica de manera general se presentó el trabajo en diversos eventos científicos. Los resultados obtenidos demuestran la novedad y pertinencia del tema, además las observaciones realizadas contribuyeron a elevar la calidad de la propuesta.

La presente investigación fue presentada en varios eventos los cuales son mencionados a continuación:

- El 28 de marzo del presente año se le otorgó el premio de 3er Lugar en la Copa de Ingeniería y Gestión de Software.
- El 9 de mayo se le otorgó el premio de Relevante en la XII Jornada Científica Estudiantil a nivel de Facultad.
- El 22 de mayo se le otorgó el premio de Destacado en la XII Jornada Científica Estudiantil a nivel de Universidad.

Para más detalle (Ver anexo 1), (Ver anexo 2), (Ver anexo 3) respectivamente.

3.5. Conclusión

En este capítulo se llevó a cabo la validación de la propuesta, para ello se realizaron entrevistas a especialistas de la línea, además de métricas donde se indicaron indicadores (preguntas). Se alcanzaron resultados favorables, obteniéndose una evolución de Bien. Todos los especialistas interrogados consideraron en que puede ser efectiva la aplicación del modelo propuesto. En relación a las opiniones, se puede concluir que la propuesta fue validada por la totalidad de los especialistas seleccionados, los cuales arrojaron que la propuesta reduce el tiempo de desarrollo de las nuevas personalizaciones a realizar en la línea. Al analizar los resultados obtenidos por las pruebas, se puede decir que el modelo de variabilidad cumple con la calidad requerida.

CONCLUSIONES GENERALES

Concluido el modelo de variabilidad de Xedro-ERP, objetivo de la presente investigación, se arribó a los siguientes resultados:

- El estudio de los conceptos fundamentales relacionados con el dominio del problema, así como la descripción de elementos importantes para el modelado de la variabilidad para una LPS, permitieron tener un mayor entendimiento del tema y se establecieron las bases para saber cómo realizar el modelo resultante.
- Se escogió para el modelado las dimensiones variabilidad en arquitectura, componentes y características, presentado como modelo de variabilidad.
- La realización de la validación del modelo de variabilidad, demostró que la aplicación del modelo permite mejorar el tiempo de desarrollo de las personalizaciones a realizar en Xedro-ERP. Lo que implica desde el punto de vista teórico el cumplimiento de la idea a defender trazada.

RECOMENDACIONES

Este trabajo fin de carrera abre varías líneas de investigación relacionadas con el modelo de variabilidad en una LPS. En particular, se recomienda para esta investigación:

- Aplicar el modelo de variabilidad a todos los productos de la línea Xedro-ERP con el objetivo de comprobar la eficiencia de la misma.
- Realizar el estudio sobre FAMA Framework, pues este provee funcionalidades que permite la gestión de variabilidad.
- Crear una herramienta informática que permita hacer el análisis de variabilidad usando el FAMA Framework, para lograr una mayor eficiencia en la aplicación del modelo.

REFERENCIAS BIBLIOGRÁFICAS

Andrews, Anneliese y Sudipto, Ghosh. 2002. *A model for Understandig Software Componets.* 2002.

Bass, L., P. Clements, and R. Kazman. 2003. *Software Architecture in Practice.* Addison Wesley : Segunda Edición, 2003.

Benavides, David. 2008. *Fama framework In software product line conference tool demostrations.* 2008.

Bosch, J. 2000. *"Design & Use of Software Architectures. Adopting and Evolving a Product-Line Approach ".* Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. : Addison-Wesley. 2000, 2000.

Brooks., Frederick P. 1987. *No silver bullet: Essence and accidents of software engineering. .* 1987. IEEE Computer, 20: 10–19..

Carneiro, Fabricia. 2011. *Análisis automático de LPS usando distintos modelos de variabilidad.* 2011.

Clements, P. 2002. *Documenting Software Architectures: Views and Beyond.* s.l. : Addison-Wesley., 2002.

Clements, P. y Northrop, L. 2001. *Software Product Lines: Practices and Patterns.* s.l. : Addison-Wesley, 2001.

Czarnecki K., Helsen S., and Eisenecker U.,. 2005. *Staged Configuration Through Specialization Multi-Level Configuration of Feature Models. Software Process Improvement and Practice, special issue on "Software Variability: Process and Management".* 2005.

Díaz, Óscar y Salva., Trujillo. 2010. *Líneas de Producto de Software.* Facultad de Informática, Universidad del País Vasco : 2º edición, 2010. Editorial Ra-Ma.

ERP. 2013. *Boletín del Centro de Informatización de la Gestión de Entidades.* 2013. 3 Edición.

Espinosa, María Eugenia Cabello, Pulido, Jorge Rafael Gutiérrez y Lizárraga, Mary Carmen Delgado. 2011. *Modelado de variabilidad en una Línea de Producto de Software.* 2011.

FAMA. 2010. Famats. *Famats.* [En línea] 2010. [Citado el: 15 de abril de 2014.] http://code.google.com/p/famats/.

Griss. M., J. Favaro, M. d'Alessandro. 1998. *Integrating feature modeling with the RSEB.* Proceedings of the Fifth International Conference on Software Reuse, Vancouver, BC, Canada : s.n., 1998.

Hofmeister, C., Nord, R. y D., Soni. 2000. *Applied Software Architecture.* 2000. Addison Wesley..

Jiménez Méndez, Alberto. *Gestión de la Variabilidad en Líneas de Producto Software.*

Jones., Linda M. Northrop y Lawrence G. 2008 . *Introduction to software product line adoption.* Washington, DC, USA : IEEE Computer Society., 2008 .

Kang K., Kim S., Lee J., Kim K., and Shin E.,. 1998. *FORM: A Feature Oriented Reuse Method with Domain Specific Reference Architectures.* s.l. : Annals of Software Engineering, Vol.5, 1998.

Kang, K. C., S. Cohen, J. Hess, W. Nowak, and S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study. .* Software Engineering Institute (Carnegie Mellon), Pittsburgh : s.n., 1990. Technical Report, CMU/SEI-90-TR-21.

Krueger, Charles W. 2006. *New methods in software product line practice. Communications of the ACM.* 2006. ISSN 0001-0785.

Montilva, J. A. 2006. *Desarrollo de Software Basado en Líneas de Productos Software. IEE Computer Society Region9 Capitulo Argentina Programa DVP.* 2006.

Northrop., Paul Clements. Linda. 2007a. A Framework for Software Product Line Practice. [En línea] 2007a. http://www.sei.cmu.edu/productlines/frame_report/coreADA.htm. Version 5.0.

Pohl K., Bockle G., and Van der Linden F. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques.* 2005.

Pohl., Günter Halmans and Klaus. 2003. *Communicating the variability of a software-product family to customers.* 2003. volume 2.

RAE. Real Academia Española. *Real Academia Española.* [En línea] [Citado el: 20 de mayo de 2014.] www.rae.es.

Sánchez, Raúl Puerta. 2011. *Soporte a la trazabilidad en el desarrollo de Líneas de Producto Software.* Madrid : s.n., 2011.

Silvana, Cinthya, Carhuayo Bendezu, Mayra Yahaira y Ramírez, Ramírez. 2012. *Aplicación de la metodología de los sistemas blandos en el modelado del reciclaje de la basura electrónica.* Lima,Perú : s.n., 2012.

Sinnema, M. 2004. *COVAMOF: A Framework for Modeling Variability in Software Product Families.* 2004.

Sistema Integral de Gestión CEDRUX. **2013.** Versión 1.0, Cuba : s.n., 2013.

Sodhi, J. y Sodhi, P. 1999. *Software reuse: Domain analysis and design process. .* New York : McGraw-Hill : s.n., 1999.

Trinidad y otros., Pablo y. 2008. *FAMA framework.* s.l. : IEEE Computer Society, 2008. In SPLC.

Tussén, Cecilia Dupotey y Pérez, Eduardo René Medina. 2013. *Estrategia para la gestión de la variabilidad de los componentes desarrollados sobre la plataforma Atlas.* Cuba : s.n., 2013.

Van Group J., and BoschJ. 2002. *Design Erosion: Problems and Causes, Journal of Systems & Software.* 2002.

Wilson, Brian. 1993. *SISTEMAS: Conceptos Metodología y Aplicaciones.* 1993.

Zubrow, D. and G. Chastek. *Measures for software production lines, Software Engineering Institute.* Carnegie Mellon University : s.n.

BIBLIOGRAFÍA CONSULTADA

A. Laguna, Miguel., Gonzalez, Bruno., Lopez, Oscar. *Gestión de la Variabilidad en Líneas de Productos.*

B. Magro, J. Garbajosa, and J. Pérez. 2009. *"Applied Software Product Line Engineering".* Taylor and Francis : Development of a Software Product Line for Validation Environments Software, 2009.

Bachmann, Felix y Clements, Paul C. 2005. *Variability in Software Product Lines.* s.l. : TECHNICAL REPORT CMU/SEI-2005-TR-012 ESC-TR-2005-012, 2005.

Bass, L., Clements, P., Donohoe, P ., McGregor, J. and Northrop, L. 2000. *" Fourth Product Line Practice Workshop Report".* s.l. : Technical Report CMU/SEI-2000-TR-002 (ESC-TR-2000-002), Software Engineering Institute, 2000.

Bass., Felix Bachmann and Len. 2001. *Managing Variability inSoftware Archi-tecture.* 2001.

Bosch, Juan., Florijn, Gert., Greefhorst, Danny., Kuusela, Juha., Obbink, Henk., Pohl, Klaus. *"Variability Issues in Software Product Lines".*

Eriksson, M., Borstler, J. y borg, K. 2005. *The pluss approach-domain modeling with features, use case and use case realizations. In software Product Lines.* s.l. : Springer-Verlag, 2005.

Felix Bachmann, Paul C. Clements. 2005. *Variability in Software Product Lines.* 2005.

IEEE, Standard Asotiation. 2000. *IEEE Recommended Practice for Architectural Description of Software- Intensive Systems.* 2000. IEEE Product No SH94869-TBR.

Islas, J. 2010. *Métricas para una Dirección de Proyectos Exitosa.* Guadalajara, México. : s.n., 2010.

Jabbedari., Gholami Z. N. Modiri and S. 2010. *Monitoring Software Product Process Metrics.* 2010. Volumen XXX.

K. Lee, K. C. Kang, and J. Lee. 2002. *Concepts and guidelines offeature modeling for product line software engineering.* 2002. Lecture Notes in Computer Science, 2319.

L. Hotz, K. Wolter, T. Krebs, S. Deelstra, M. Sinnema, J. Nijhuism, y J. MacGregor. 2006. *Configuration in Industrial Product Families. .* 2006. ISBN 978-1-58603-641-6.

Len Bass, Paul Clements, Sholom Cohen, Linda Northrop, y James Withey. 1997. *Product line practice workshop report. .* Software Engineering Institute, Carnegie Mellon

University, Pittsburgh, Pennsylvania 15213 : s.n., 1997. Reporte técnico CMU/SEI-97-TR-003.

Liu, D., Mei, H. 2004. *From requirements to software architecture: A feature oriented mapping approach.* 2004. In Proc. of the 2nd Int. Software Requirementsto Architectures Workshop, ACM Press.

Marco Sinnema, Onno De Graaf, y Jan Bosch. 2004. *Tool support for COVAMOF. EnInternational Workshop on Software Variability Management for Product Derivation.* 2004.

Mc-Graw-Hill. Pressman, R. 2002. *Ingenieria del software: un enfoque práctico.* 2002.

Pfleeger, Shari. 2002. *Ingeniería del Software. Teoría y Práctica.* 2002.

PMI, P.M.I. 2004. *Guía de los Fundamentos de la Dirección de Proyectos.* 2004. Tercera Edición.

Pohl., Kim Lauenroth y Klaus. 2005. *Principles of variability.* 2005. ISBN 3-540-24372-0..

Pressman, R. S. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.

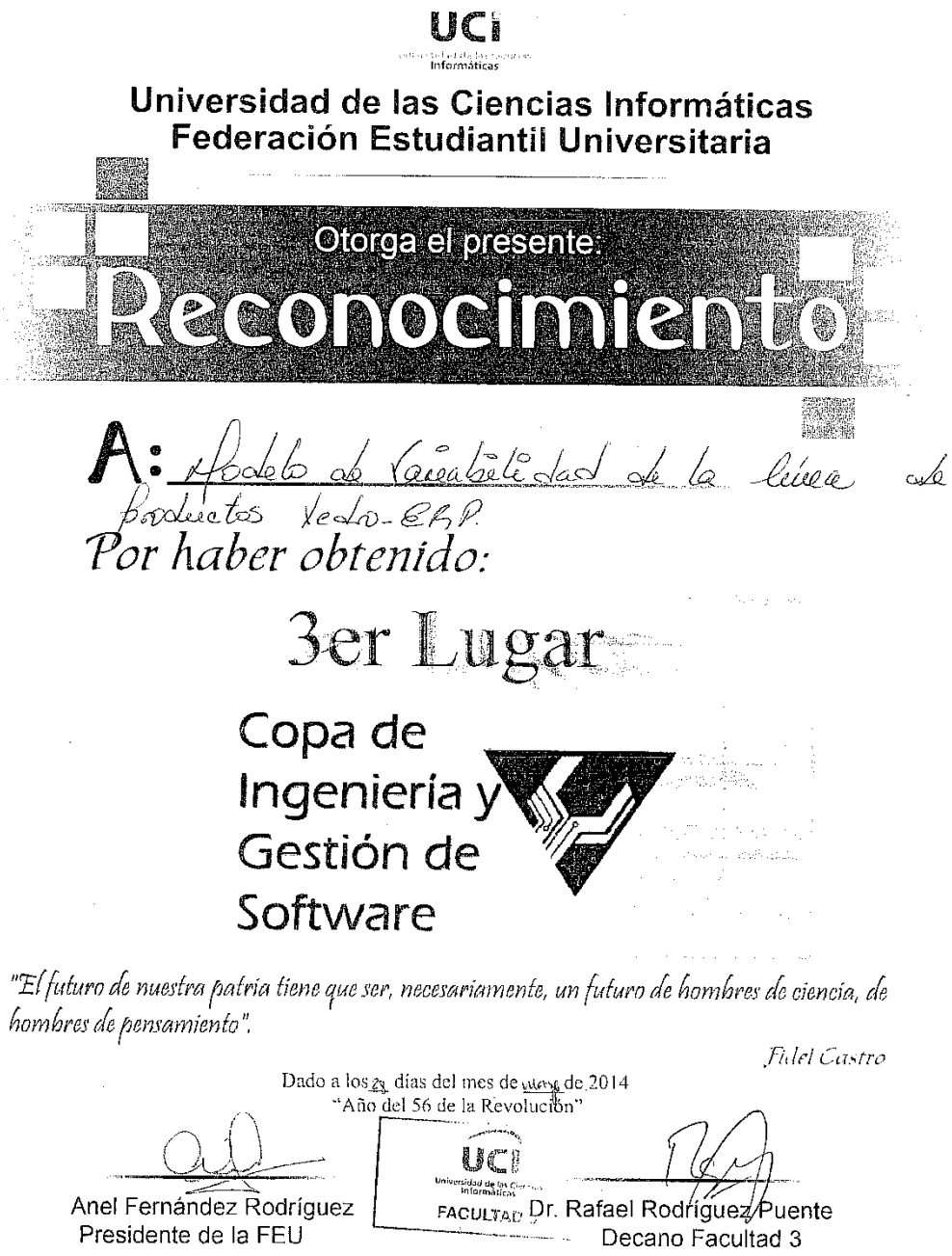
SEI, Software Engineering Institute. 2007. A Framework for Software Product Lines Practice. [En línea] 2007. http://www.sei.cmu.edu..

Sinnema., Sijbren Keimpe Deelstra y Marco. 2008. *Managing the Complexity of Variability in Software Product Families.* 2008.

Szyperski, Clemens. 1998. *Component Software Component Software Beyond Object-Oriented Programming.* 1998.

ANEXOS

Anexo # 1: Premio de 3er Lugar que se obtuvo en la Copa de Ingeniería y Gestión de Software.



UCI
Universidad de las Ciencias Informáticas

Universidad de las Ciencias Informáticas
Federación Estudiantil Universitaria

Otorga el presente:

Reconocimiento

A: Modelo de Variabilidad de la línea de productos XeLo-ERP.

Por haber obtenido:

3er Lugar

Copa de Ingeniería y Gestión de Software

"El futuro de nuestra patria tiene que ser, necesariamente, un futuro de hombres de ciencia, de hombres de pensamiento".

Fidel Castro

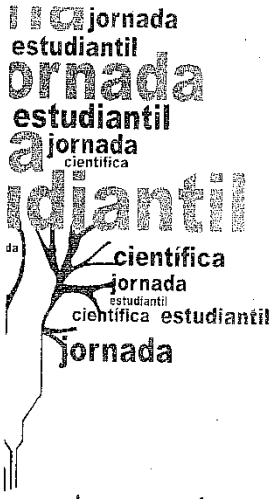
Dado a los 23 días del mes de mayo de 2014
"Año del 56 de la Revolución"

Anel Fernández Rodríguez
Presidente de la FEU

UCI
Universidad de las Ciencias Informáticas
FACULTAD

Dr. Rafael Rodríguez Puente
Decano Facultad 3

Anexo # 2: Premio de Relevante que se obtuvo en la XII Jornada Científica Estudiantil a nivel de Facultad.



UCI Universidad de las Ciencias Informáticas

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FEDERACIÓN ESTUDIANTIL UNIVERSITARIA

otorga el presente

RECONOCIMIENTO

a: Zaima Guerra Sardenes.

por obtener: Relevante

con el trabajo: Modelo de mantenibilidad de la línea de Productos XEDAO - ERP.

En la XII edición de la JORNADA CIENTÍFICA ESTUDIANTIL en la FACULTAD 3.

"La tecnología es el reflejo del fanatismo del hombre por sobrevivir"

DADO A LOS 9 DÍAS DEL MES 5 DEL 2014

"AÑO 56 DE LA REVOLUCIÓN"

ERNESTO ORTIZ MUÑOZ
PRESIDENTE DE LA FEU

UCI Universidad de las Ciencias Informáticas
FACULTAD 3

DrC. RAFAEL RODRÍGUEZ PUENTE
DECANO

Anexo # 3: Premio de Destacado que se obtuvo en la XII Jornada Científica Estudiantil a nivel de Universidad.

UCI

Universidad de las Ciencias Informáticas
Federación Estudiantil Universitaria

Otorgan el presente:

RECONOCIMIENTO

A: Modelo de variabilidad de la línea de productos Xedro-ER.

por obtener la condición de

DESTACADO

en la XII Edición de la Jornada Científica Estudiantil.

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica"

Aristóteles

Dado a los 22 días del mes de mayo de 2014.

Ailyn Febles Estrada
Vicerrectora de Investigación y Postgrado

Víctor Gabriel González Cardoso
Presidente de la FEU

GLOSARIO DE TÉRMINOS

Arquitectura: La arquitectura es la parte fundamental de un sistema incorporada en sus componentes, sus relaciones con otros, y los principios que guían su diseño y evolución.

Capital Humano: El subsistema de Capital Humano de Xedro-ERP abarca un grupo de procesos que interactúan entre sí como son: Organización del trabajo, Selección e integración y Administración de Capital Humano.

Componente: Un componente es una parte no trivial, casi independiente, y reemplazable de un sistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces.

Contabilidad General: El subsistema de Contabilidad General de Xedro-ERP permite toda la gestión de la información contable, configuración de las cuentas, emisión de comprobantes de operaciones en moneda contable y original, registro de asientos tipos de todos los subsistemas al estar concebido para recibir y procesar la información del resto de los subsistemas a través de los comprobantes de operaciones. De igual manera permite la emisión del mayor y los submayores de análisis de cada una de las cuentas definidas en el nomenclador de cuentas.

ERP: Enterprise Resource Planning, traducido al español (Planificación Recursos Empresariales).

Expediente de Proyecto: Nombre que se le da al resultado de la documentación en el proceso de desarrollo de software.

FAMA: *Framework_for the Automated Analysis of Feature Models*, *framework* o marco automatizado para el análisis de modelos de características.

Feature-RSEB: *Reuse-Driven Software Engineering Business*, traducido al español (Método de reutilización orientado a características con referencia específica al dominio), para el modelado de variabilidad, añade una nueva relación entre una característica padre y una característica hija.

FODA: *Feature- Oriented Domain Analysis*, traducido al español (Análisis del dominio orientado a características). Primer modelo característico para extraer las similitudes y variabilidades de un dominio.

FORM: *Feature Oriented Reuse Method*, traducido al español (Método de Reuso Orientado a Características) es una extensión del método FODA para el modelado de la variabilidad en otras fases del desarrollo.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Gestión Financiera: El subsistema de Gestión Financiera de Xedro-ERP busca la estandarización de los procesos contables en las empresas así como la integración de las actividades de Caja, Banco y Cobros y Pagos, teniendo una correspondencia y adaptabilidad a las nuevas legislaciones financieras e informáticas.

IoC: Son las siglas de Inversion of Control, traducido al español (Inversión de Control). Mueve la responsabilidad de realizar las tareas al inferior del framework y fuera del código de la aplicación.

Reutilización: Acción y efecto de reutilizar.

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación. Equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados *hardware*. Los componentes lógicos incluyen, entre muchos otros, las aplicaciones informáticas; tales como el procesador de textos, que permite al usuario realizar todas las tareas concernientes a la edición de textos; el *software* de sistema, tal como el sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando también la interacción entre los componentes físicos y el resto de las aplicaciones, y proporcionando una interfaz para el usuario.

Subsistema: Cada uno de los componentes principales de un sistema que este dividido en componentes. Cada subsistema abarca aspectos del sistema que comparten alguna propiedad común.

Validación: Comprobación de que se está construyendo el producto correcto.

Variabilidad: Cualidad de las cosas que tiende a cambiar o transformarse según la Real Academia Española.

Xedro-ERP: Es una solución informática que se construye adaptado al control de los recursos empresariales de la entidad.