UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título: "Modelo de desarrollo de software basado en Líneas de Producción de Software y técnicas de desarrollo ágil"

Trabajo final presentado en opción al título de
Máster en Gestión de Proyectos Informáticos

Autor: Ing. Dorisbel Muro Fumero

Tutor: MSc: Roberto González Dum

Ciudad de la Habana, junio de 2011

RESUMEN

La presente investigación realiza un estudio del estado del arte de los diferentes modelos de desarrollo de software, incursionando en los modelos de desarrollo de Líneas de Producción de Software (LPS), específicamente en el modelo desarrollado por el Software Engineering Institute (Instituto de Ingeniería de Software SEI). Por otro lado se analizan las metodologías de desarrollo de software, especialmente SCRUM. El modelo centra sus postulados en los conceptos de Ingeniería de Dominios, Ingeniería de Aplicaciones, Administración de la LPS y Arquitectura, como disciplina que se le adhiere al modelo con el objetivo de lograr una abstracción arquitectónica de la solución antes de los desarrollos. El modelo propone cinco fases fundamentales: Concepción, Conceptualización Arquitectónica, Ingeniería de Dominios, Ingeniería de Aplicaciones y Empaquetado y Despliegue, todas ellas gestionadas por la metodología de desarrollo SCRUM y los principales postulados de gestión de proyectos. El modelo toma como punto de partida las experiencias adquiridas en las empresas nacionales de desarrollo de software SICS y DESOFT; presentando los resultados de su aplicación en función de la agilidad, productividad y reutilización que el modelo proporciona a las empresas, el nivel de cubrimiento de los elementos fundamentes que deben contemplar las LPS por los entornos objetivos y la justificación del uso de metodologías ágiles en dichos entornos.

Palabras claves: Modelo de desarrollo de software; Línea de producción de software, SCRUM.

ABSTRACT

This research conducts a state of the art of different software development models, moving into the development models of Software Production Line (LPS), specifically on the model developed by the Software Engineering Institute (SEI). On the other hand examines the software development methodologies, especially SCRUM. The model focuses on the concepts of domain engineering, application engineering, administration of LPS and Architecture, a discipline that you add to the model with the goal of an architectural abstraction of the solution before the developments. The model proposes five main phases: Design, Conceptualization Architectural, Engineering, Domain Engineering and Application Packaging and Deployment, all managed by the SCRUM development methodology and the main concepts of project management. The model takes as its starting point the experiences in national development companies software DESOFT and SICS, presenting the results of its application in terms of agility, productivity and reuse that the model gives companies, the level of coverage of the fundamental elements which must take into consideration the environments LPS objectives and the reason for using agile methodologies in these areas.
Keywords: Software development model, Software production line, SCRUM.

ÍNDICE

INTRODUCCIÓN

En la actualidad la globalización exige flexibilidad a las empresas de desarrollo de software para mantener su auge en los próximos años. Debido a esto el ambiente de negocios ha cambiado y continuará variando en forma constante, por lo que las empresas deben desarrollar la capacidad de respuesta rápida ante nuevas oportunidades de negocio y retos competitivos al enfrentar cambios en las condiciones de mercado. De esta manera la mayoría de las empresas han venido creciendo y mejorando sus procesos, lo que ha conllevado a la aplicación de nuevas prácticas en la industria de software, siendo necesario reflexionar sobre el grado de adaptación de los modelos de desarrollo de software utilizados hasta el momento (1).

Las metodologías ágiles de desarrollo de software, en comparación con las metodologías convencionales, perciben cada respuesta al cambio como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente, considerando la gestión de proyecto como un aspecto inherente al propio proceso de desarrollo de software. Como parte de este proceso, la gestión de proyecto tiene que dar el paso de evolución apropiado para desarrollar nuevos productos y servicios en estos sectores y adaptarse a los cambios del entorno (2).

Las metas principales de la Industria Cubana del Software radican en: informatizar la sociedad, elevar los niveles de desarrollo de software y exportar productos de software. En la Industria Cubana del Software, a diferencia del resto de los países de América Latina, se hace más complejo el acceso al mercado internacional producto al bloqueo económico sostenido por parte de Estados Unidos, lo que la provoca invertir varias veces más recursos al tener que recurrir a mercados muy distantes. Se requiere acelerar estas metas para clasificar entre los primeros países de América Latina que obtienen resultados en esta industria y acortar la brecha tecnológica entre los países subdesarrollados y las economías de los países del primer mundo, contribuyendo así al desarrollo socio-económico del país. Por esta razón existe necesidad de acortar los ciclos productivos y mantener un elevado porciento de efectividad en las respuestas a los clientes de cada entorno de negocio. A pesar de los elementos externos que afectan la industria cubana del software y de la precocidad de esta se puede aprovechar más las potencialidades que posee Cuba. Donde se evidencia una favorable cultura educativa, la profesionalidad de los recursos humanos y las iniciativas que se han llevado a acabo para impulsar el desarrollo de la informática, como lo constituye: la creación de la Universidad de las Ciencias Informáticas (UCI), la creación de los Joven Club de Computación y la potencialización del Ministerio de informáticas y Comunicaciones (MIC). Se puede además encaminar esfuerzos para la realización de una estrategia general que abarque no solo la informatización del país, sino que contemple toda la industria como un factor clave para el desarrollo de la nación (3).

La presente investigación tiene su aplicación práctica en dos entornos de negocio diferentes: la

empresa DESOFT, puntera en la producción cubana de software, cuyo objetivo fundamental es proporcionar soluciones integrales en tecnologías de la información para la informatización de la Sociedad Cubana y la empresa SICS (Servicios Informáticos de Consultoría y Sistemas) que tiene la misión de proporcionar soluciones informáticas para el Ministerio del Transporte. A partir del diagnóstico organizacional realizado a las empresas de carácter nacional DESOFT y SICS se evidencia que: los equipos de desarrollo cuentan con pocas personas para llevar a cabo todo el proceso de trabajo. Estos equipos están integrados en parte por técnicos medios en informática (que no reciben una formación completa de los procesos ingenieriles de desarrollo de software), ingenieros informáticos recién graduados (se encuentran en períodos de adiestramiento y tienen poca experiencia en el desarrollo de software) e ingenieros y licenciados con varios años de experiencia (representan una minoría, muchos están desactualizados con respecto a las tecnología y tendencias actuales de desarrollo de software). Esta situación trae como consecuencia que los equipos presenten dificultades en aplicar modelos de desarrollo de software novedosos, que permitan agrupar esfuerzo hacia metas comunes. Los equipos de trabajo también presentan dificultades en gestionar el conocimiento adquirido en cada experiencia, viéndose afectada la calidad del flujo de información entre los integrantes del equipo de desarrollo, así como las comunicaciones entre las empresas productora y clientes. Difícilmente se encaminan esfuerzos a realizar estudios arquitectónicos del producto que presenten una visión de la solución de software respecto al entorno de negocio que se requiera implantar, proporcionando limitaciones en los elementos técnicos necesarios para la construcción y el cumplimento de las métricas requeridas que den calidad al éxito del producto. No existe una cultura de producción de software en la que se realicen estudios por parte de equipos multidisciplinarios encaminados a la creación de un producto orientado a un mercado amplio. El tipo de solución que se desarrolla para todos los renglones sociales es predominantemente software de gestión, realizado a la medida para un cliente específico; lo que trae consigo una baja capacidad de reutilización entre soluciones y proyectos. Otro elemento que contribuye a duplicar esfuerzos y tiempo es la pobre estandarización e implantación de tendencias industriales en los modelos de procesos, así como una poca formalización de procesos y modelos productivos, tanto para la gestión de proyecto como para el desarrollo del software. Lo anteriormente planteado trae como consecuencia además, que muchas veces no exista documentación que describa los elementos esenciales del proceso de desarrollo de software que permita realizar análisis retroactivo del mismo. En otras ocasiones la documentación es basta, dejándose de especificar los elementos fundamentales del proceso. Se presentan dificultades en adaptar las soluciones informáticas a los cambios que exigen los clientes y los entornos de negocio; dilatando el tiempo de respuesta, lo que afecta directamente en la calidad y el costo de los productos informáticos. Otro aspecto que dilata el tiempo de respuesta a los clientes es la carencia de mecanismo de reutilización que permitan implementar soluciones solo una vez, pudiéndose mejorar las mismas, socializarlas y adaptarlas a

los entorno clientes. Las empresas no siempre pueden cumplir con los planes establecidos, por lo que no obtienen todos los ingresos esperados, afectando a sus objetos sociales y a sus objetivos, limitándose además de poder dar una respuesta rápida y con alto porciento de efectividad, como necesita la sociedad cubana. Analizada toda la problemática anterior se propone como **Problema de investigación:** ¿Cómo mejorar los procesos ingenieriles de los modelos desarrollo de software en las empresas nacionales SICS y DESOFT que contribuya a aumentar la agilidad, reutilización y productividad en la obtención de nuevos productos y servicios?

Objeto de estudio: modelos de desarrollo de software.

Objetivo general: construir un modelo de desarrollo de software basado en LPS que contribuya al incremento de la agilidad, reutilización y productividad del proceso productivo de desarrollo de software en las empresas nacionales SICS y DESOFT.

Campo de acción: modelos de desarrollo de software basados en LPS.

Objetivos específicos:

1. Realizar estudio del estado del arte de los principales enfoques y modelos de procesos y metodologías de desarrollo de software para delimitar el marco teórico de la investigación.
2. Definir un modelo de procesos de desarrollo de software basado en LPS enfocado en los aspectos ingenieriles para las empresas nacionales SICS y DESOFT.
3. Aplicar y evaluar el modelo de procesos de desarrollo de software basado en LPS en las empresas SICS y DESOFT en dominios que no involucren desarrollo para hardware específico ni complejos ciclos de aprendizaje.

Posibles resultados:

1. Modelo de procesos de desarrollo de software basado en LPS, con contribución al incremento de la agilidad, reutilización y productividad del proceso productivo de desarrollo de software en las empresas nacionales SICS y DESOFT.
2. Expediente con la especificación de la taxonomía de los elementos que documentan el modelo, ajustado con la metodología ágil de desarrollo SCRUM.

Hipótesis: Si se construye e implanta un modelo de desarrollo de software basado en LPS en las empresas nacionales SICS y DESOFT, se contribuirá a aumentar la agilidad, reutilización y productividad en la obtención de nuevos productos y servicios.

Variables dependientes: *Agilidad* (para el establecimiento de los indicadores se tuvieron en cuenta los principios del manifiesto ágil (4)), *Reutilización* (para el establecimiento de los indicadores se tuvo en cuenta los principios de reutilización que plantean los modelos de LPS (5) (6)) y *Productividad* (para el establecimiento de los indicadores se tuvo en cuenta los mecanismos de medición de productividad señalados por las empresas nacionales SICS y DESOFT).

Variable independiente: Modelo de desarrollo de Software.

Tabla 1 Operacionalización de las Variables dependientes.

Variable	Dimensión	Indicador	Concepto	UM
Agilidad	Documentación precisa.	Cantidad de artefactos.	Número de documentos sin pérdida de información.	Número
	Nivel Autogestión del equipo.	Autonomía	Libertad para tomar decisiones.	Alto, Medio, Bajo
		Auto-enriquecimiento	Transferencia de conocimiento.	Alto, Medio, Bajo
Reutilización	Activos reutilizados.	Cantidad de componentes reutilizados.	Número de activos que se pueden emplear en diferentes soluciones.	Número
	Mecanismo de reutilización.	Cantidad de mecanismos de reutilización establecidos.	Conceptos que se logren establecer del modelo, en las empresas, que constituyan elementos de reutilización.	Número
Productividad	Ciclo de vida del producto.	Tiempo de terminación del producto.	Tiempo que demora el producto en desarrollarse y entregarse al cliente.	Meses
	Obtención de nuevos productos.	Cantidad de productos.	Capacidad de obtener nuevos productos en menos tiempo.	Número
	Eficiencia del desarrollo.	Eficiencia	(Cantidad productos * Índice complejidad)/ (Cantidad de personas * Ciclo vida promedio)	Número

Tabla 2 Operacionalización de la Variables independiente.

Variable	Indicador	Sub-indicadores	UM
Modelo de desarrollo de software basado en Líneas de Producción de Software y técnicas de desarrollo ágil.	Calidad del modelo.	Calidad de la documentación.	Poco adecuado (10) Adecuado (20) Muy adecuado (30)
		Capacidad de ser generalizado.	Alto, Medio, Bajo
		Completitud (áreas del conocimiento que abarca)	Poco adecuado Adecuado Muy adecuado

La unidad de estudio radica en el proceso de organización del desarrollo de proyectos de software en las empresas del entorno cubano SICS y DESOFT.

Población: la población está compuesta por los productos de los dominios de desarrollo de las empresas software que no involucran desarrollo para hardware específico ni complejos ciclos de

aprendizaje en las empresas SICS del Ministerio del Transporte que comprenden seis soluciones a la medida y de la Gerencia de Desarrollo de la División Ciudad Habana DESOFT que comprenden 22 soluciones a la medida.

Estos comparten como característica comunes las tecnologías de desarrollo, la tipología de solución de software, enmarcada en el dominio de los sistemas de gestión, dimensiones pequeñas y medianas, características de las personas involucradas, complejidad de la solución y tamaño de los equipos. Los mismos presentan como principal diferencia las tecnologías de desarrollo utilizadas, que son variadas, sin uso de marcos de trabajo tecnológicos y basadas fundamentalmente en plataformas .Net, PHP y Java.

Muestra: se seleccionó el 100 % de los proyectos de la población de ellos 2 de los proyectos existentes del SICS y 16 de los proyectos DESOFT.

Métodos de investigación utilizados

Empíricos: en el presente trabajo se utilizan métodos empíricos al observar y estudiar las características fundamentales y las relaciones esenciales del objeto de estudio en el estado del arte que materializa el capítulo 1. Entre las técnicas empleadas están: la entrevista y la encuesta, realizadas antes de implantar el modelo y luego de haberlo implantado, con el objetivo de analizar los indicadores de las variables a medir en la investigación (7) (8).

Teóricos*:* en el presente trabajo se asumen conceptos con interpretación, comprensión y explicación de los datos, hechos o informaciones recogidas mediante los métodos empíricos, que estudiaron los modelos de desarrollo de software basados en LPS y las prácticas ágiles de desarrollo, conduciendo a descubrir y revelar la esencia del objeto de estudio y sus relaciones. Entre los procesos del pensamiento usados están:

Análisis- síntesis: al analizar los principales conceptos canones de la investigación y sinterizar los elementos fundamentales de ellos.

Histórico-lógico: para la evaluación del estado del arte, tomado como referencia para la investigación y como punto de comparación de los resultados alcanzados. Entre los métodos lógicos usados en el presente trabajo están:

Hipotético-deductivo: para el análisis y la definición de la hipótesis de la investigación que será verificada o valorada en función del estudio de elementos más particulares, de menor nivel de generalidad.

Sistémico: para determinar los elementos de los modelos de desarrollo de software basados en el paradigma LPS y las posibles relaciones existentes entre ellos, hasta lograr un funcionamiento integrado de todos estos elementos (7) (8).

Estructura:

Capítulo 1: Fundamentación Teórica.

Este capítulo contiene los fundamentos teóricos para entender el problema a solucionar. Abarca los conceptos fundamentales del objeto de estudio planteado como es: el estudio del estado del arte de los modelos de desarrollo de software, los modelos de LPS, específicamente el modelo propuesto por el SEI y las metodologías de desarrollo de software tanto robustas como ágiles realizando un énfasis minucioso en la metodología SCRUM.

Capítulo 2: Desarrollo de la solución.

Contiene la propuesta del modelo de desarrollo de software fundamentando los elementos claves del mismo. Se describen cada una de las fases (Concepción, Conceptualización Arquitectónica, Ingeniería de Dominios, Ingeniería de Aplicaciones y Empaquetado y Distribución), así como también se describen los procesos de Administración de la LPS instanciando la metodología ágil de desarrollo de software SCRUM.

Capítulo 3: Evaluación de la solución.

Se realiza la evaluación de la propuesta con el objetivo de valorar la solución planteada. Se analiza la materialización del paradigma LPS a parir de las experiencias adquiridas en la implantación del modelo en las empresas del entorno cubano de desarrollo de software SICS y DESOFT. Se comparan los resultados obtenidos del diagnóstico realizado a las empresas antes de aplicar el modelo y después de haberlo aplicado, analizando el resultado de las variables señaladas en el diseño teórico. Se realiza un análisis de las empresas estudiadas respecto a los criterios de Bohem y Turner.

CAPÍTULO 1 MARCO TEÓRICO

Introducción

En el presente capítulo se abordan los elementos teóricos de los principales canones de los modelos de desarrollo de software tradicionales, así como los principales modelos europeos y americanos de Línea de Producción de Software (LPS), abordando específicamente el perteneciente al Software Engineer Institute (SEI). Por otro lado se realiza un estudio del estado del arte de las principales metodologías de desarrollo de software, realizando un mayor debate sobre SCRUM.

1.1 Modelos de procesos de desarrollo de software

Un modelo de procesos de desarrollo de software puede definirse como un conjunto de actividades que conducen a la creación de un producto de software. Existen diferentes modelos, pero comúnmente el ciclo de vida consiste en: especificación, diseño e implementación, validación y evolución del software. (9) (10) (11) Entre los modelos más comunes de encuentran:

Modelo en cascada: consiste en la realización continua del proceso de desarrollo de software. Las fases son: definición de requerimientos, diseño, implementación pruebas de unidades, integración y pruebas del sistema, funcionamiento y mantenimiento. Una fase no comienza hasta que la antecesora no termine. Permite manejar fácilmente los planes del proyecto pero tiene el inconveniente de que los usuarios no pueden ver como va quedando el sistema hasta el final del desarrollo, que casi siempre se hace bastante largo, razón por la que se hace engorrosa la tarea de darle respuesta a los cambios en los requerimientos (10) (11).

Desarrollo por prototipos: se centra en la idea de ayudar a comprender los requisitos que plantea al usuario, principalmente cuando este no tiene una idea acabada de lo que desea o el ingeniero tiene dudas sobre la variabilidad de la solución. Propicia una buena comunicación con los clientes, que pueden evaluar funcionalidades del sistema en etapas tempranas del desarrollo, lo que permite mejorar dichos prototipos, contribuyendo a la satisfacción del cliente. Esta misma ventaja puede convertirse en una debilidad, pues las planificaciones del proyecto pueden sufrir constantes cambios y en proyectos grandes pueden existir dificultades para mantener el sistema (12).

Desarrollo en espiral: la característica fundamental de este modelo es que el desarrollo de los estados se repite. Se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones, hasta que se desarrolla un sistema adecuado. Las actividades de especificación, desarrollo y validación se entrelazan en vez de separase, con una rápida retroalimentación entre éstas. Tiene lo mejor de los modelos anteriores, pues existe una buena comunicación con los clientes a partir de que combina las ventajas de prototipo y cascada. El acercamiento al sistema final paso a paso permite que los

usuarios puedan evaluar en cada iteración. El punto débil está en la dificultad para mantener las planificaciones y dimensionar el sistema globalmente (9) (11) (13). Variaciones del desarrollo en espiral (pueden realizarse simultáneamente): modelo iterativo (se repiten los flujos de trabajo para cada elemento del desarrollo del sistema) y modelo incremental (en cada iteración del desarrollo se le agregan nuevas funcionalidades) (9) (11) (10).

Desarrollo basado en componentes o paquetes: se basa en la adaptación de paquetes de software a partir del desarrollo de funcionalidades adicionales y configuración de parámetros. Supone una arquitectura base expresada en componentes, que constituyen la unidad de producción o producto parcial a producir, en el que están definidas las características de integración. Las fases fundamentales son: requerimientos, evaluación y selección de paquetes, ajuste y análisis de funcionalidades, diseño, adición de funcionalidades y desarrollo y por último pruebas. Permite la reutilización de soluciones, trayendo consigo la reducción de tiempo, costo y esfuerzo en el desarrollo de sistemas. Otra fortaleza es el aumento de la calidad de los componentes que se van estabilizando. Sus puntos débiles radican en dificultades para asegurar el versionamiento si los suministradores de componentes no mejoran continuamente sus productos, y aumento del costo si la adaptación de nuevos paquetes es costosa. La tendencia actual que se incluye como evolución de este paradigma es el desarrollo de marcos de trabajo y dominios de solución como elementos de madurez que tributan el desarrollo basado en componentes. (14) El SEI institucionaliza una teoría para la gestión del modelo basado en componentes y que se ha extendido a otros modelos conocidos como teoría formal de componentes COST (Commercial Off The Shelf por sus siglas en inglés) (9) (15) (16) (17) (5).

Desarrollo con técnicas de 4ta generación (Desarrollo Rápido de Aplicaciones (DRA) o Rapid Application Development RAD por sus siglas en inglés): el propósitos este modelo es el desarrollo rápido, con alto nivel de calidad y bajo costo. Las características fundamentales son: participación de los usuarios en todas las etapas, el uso de herramientas automáticas y el uso de equipos de desarrollo pequeños. Posibilita una buena comunicación con los clientes, donde pueden revisar el sistema sistemáticamente. Tiene el inconveniente que pueden existir dificultades para asegurar un equipo de trabajo de alto nivel profesional (18).

Fábrica de software: es un modelo de proceso de desarrollo de software cuya forma de trabajo es una analogía de los procesos de producción industriales, permitiendo una reducción de los niveles de incertidumbre en el desarrollo y mantención de proyectos de software. Se basa en la configuración de herramientas extensibles, contenidos y procesos estandarizados y mejorables continuamente, para automatizar el desarrollo y mantenimiento de variantes de un producto arquetípico, mediante la adaptación, ensamblaje y configuración de componentes basados en marcos de trabajo (19). Resumiendo las principales actividades que promueven este modelo son: construcción de familias de software similar, ensamblado de componentes, desarrollo de

lenguajes de modelado y herramientas específicas para el dominio y uso de una planificación basada en restricciones y guía activa (20). Entre los modelos más conocidos se encuentran:

- Modelo basado en la norma ISO 9001 (International Organization for Standardization por sus siglas en inglés) y CMMI (Capability Maturity Model Integration por sus siglas en inglés) (21) (22).
- Modelo Eureka (23) (22).
- Modelo Clasificatorio (24) (22).
- Modelo propuesto por Basili (25) (22).
- Modelo Replicable (22) (26).

Líneas de producción de software: diferentes autores plantean el concepto de LPS.

El Instituto de Ingeniería de Software (SEI) define una LPS basándose en el concepto planteado por Paul Clements y Linda Northrop en su libro Software Product Lines Practices and Patterns: "conjunto de sistemas de software compartiendo características comunes y administradas que satisface las necesidades específicas de un segmento de mercado particular o misión que son desarrolladas de manera prescrita a partir de un conjunto común de elementos claves" (27) (5).

Otro concepto sobre LPS es el enunciado por Charles W. Krueger: "se refieren a técnicas de ingeniería para crear un portafolio de sistemas de software similares, a partir de un conjunto compartido de activos de software usando un medio común de producción" (28).

Hassan Gomma también dio a conocer su visión sobre las LPS cuando expresó: "consisten en una familia de sistemas de software que tienen una funcionalidad común y alguna funcionalidad variable". La funcionalidad común descansa en el uso recurrente de un conjunto común de activos reutilizables (requisitos, diseños, componentes, servicios web, etc.). Los activos son reutilizados por todos los miembros de la familia (29).

Basado en estas definiciones, el trabajo asume el concepto de LPS como: conjunto de activos de software que son reutilizados, compartiendo características comunes encaminadas al dominio de un entorno de negocio y desarrollando aspectos variables que establecen diferencias entre los productos de la familia.

La Ilustración 1 muestra la estructura básica de una LPS, la cual parte de un conjunto de activos de software pre-concebidos encaminados a un entorno de mercado, a los que se les determina sus variabilidades (decisiones de producto) según la personalización a clientes concretos, obteniendo como resultado de este proceso productos de software (30) (31).

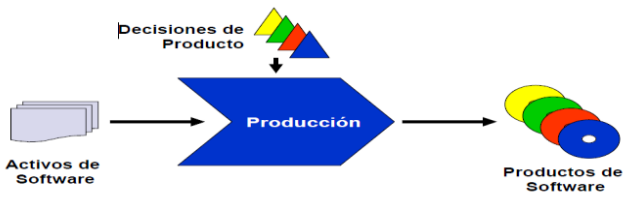


Ilustración 1 Modelo Básico de una Línea de Productos de Software (5)

El esquema que se muestra en la Ilustración 2 describe los conceptos fundamentales de una LPS. Adicionalmente las LPS manejan otro grupo de conceptos imprescindibles en su implementación. Estos conceptos son:

Arquitectura: "La arquitectura de software es la representación de alto nivel de abstracción de un sistema, constituida por las partes que lo integran, el mecanismo de integración e interrelación, restricciones y configuraciones a entornos de solución, los principios del diseño y evolución, así como el ambiente tecnológico de implantación" (32) .

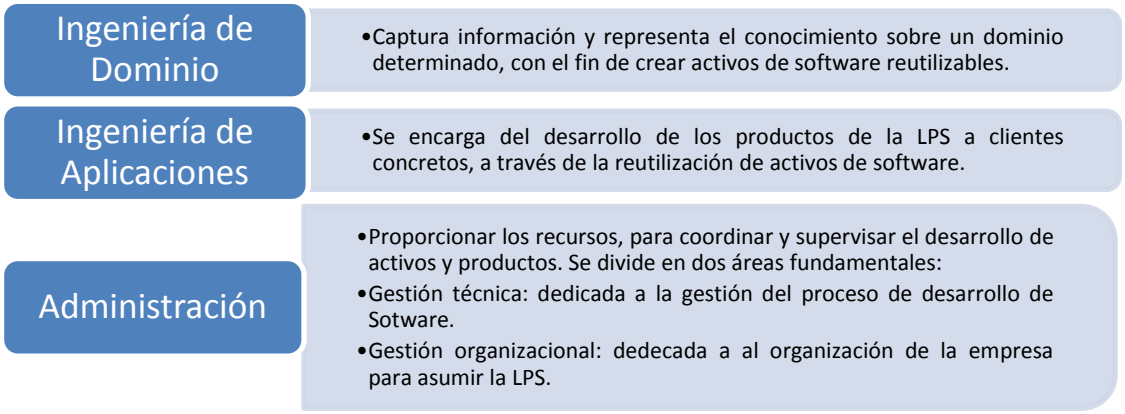


Ilustración 2 Procesos básicos de una LPS (31) (33) (5)

La arquitectura de una LPS captura los aspectos comunes y variables de una familia de productos de software. Los aspectos comunes se definen en la Arquitectura de Dominio y los aspectos variables en la Arquitectura de Aplicaciones. La Arquitectura de Dominio describe la estructura de toda la familia de productos y no solamente la de un producto particular. Debe ser instanciada cada vez que se desarrolla un producto de la línea. Mientras que la Arquitectura de Aplicaciones se encamina a la definición de los elementos de productos individuales particularizados a clientes.

Reutilización: "Reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo" (34). La reutilización no es un concepto nuevo, surge en los años 70 con la programación estructurada, evoluciona en los años 80 con la programación orientada a objetos y en los años 90 surge el término de componente reutilizable, dando lugar a los paradigmas de programación de componentes top-down o descendente (para reutilizar) y bottom-up o ascendente (basado en reutilización). Otro exponente de la reutilización surgido a finales de los años 90 y a principios este siglo (2000) los constituyen SOA (Service-Oriented Architecture), con la reutilización de servicios. De esta manera surgen las LPS, constituyendo un esquema de alta reutilización, pues se cuenta con una arquitectura de dominio genérica y se consolidan los activos de software comunes a la familia de productos, disminuyendo la duplicación de esfuerzos de ingeniería. Es gestionada porque cuenta con un enfoque sistemático, planificado, institucionalizado y mejorado (31) (6).

Repositorio: es el espacio digital que centra la reutilización. El repositorio en LPS es una base de

datos especializada que almacena activos de software, con toda la información que lo especifica y facilita la recuperación y el mantenimiento de estos de forma controlada. Su objetivo es asegurar la disponibilidad de activos para apoyar el desarrollo de productos de la LPS (31).

Las LPS pueden incrementar significativamente la productividad de los ingenieros de software, entendida como una reducción en el esfuerzo y el coste necesario para desarrollar, poner en marcha y mantener un conjunto de productos software similares. La Ilustración 3 muestra una comparación del enfoque basado en LPS con respecto al convencional, a partir del esfuerzo realizado por el equipo de trabajo en la obtención de nuevos productos. Evidenciando mucho menos esfuerzo en el desarrollo basado en LPS, que tiende a ser estable después de la construcción del primer producto (35).

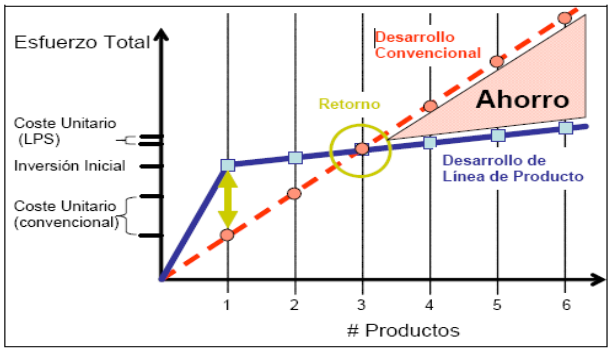


Ilustración 3 Desarrollo convencional vs Línea de producto (35)

Por otro lado las LPS proporcionan un notable impacto en la calidad. Los beneficios que las LPS aportan a la calidad se pueden medir de dos formas:

- Mediante el grado de precisión con que cada producto se ajusta a las necesidades del cliente.
- La tasa de defectos en los productos de la LPS. Aquí los beneficios se derivan de la reutilización de los elementos comunes. La continuada utilización de estos elementos a lo largo del tiempo hace que finalmente estén muy depurados/probados (35).

El uso sistemático de prácticas LPS arrojan resultados en importantes beneficios como son:

- Aumento de la calidad (hasta 10 veces).
- Reducción del coste hasta en un 60%.
- Disminución de las necesidades hasta en un 87%.
- Disminución del tiempo de respuesta en el mercado hasta en un 98%.
- Capacidad para entrar en nuevos mercados en meses, no años (6) (31).

1.2 Modelos de procesos para LPS.

1.2.1 Modelo europeos

Modelo PuLSE (Product Line Software Engineering): metodología que permite definir, construir y

mantener LPS en una variedad de contextos empresariales. Fue creado en el Fraunhofer IESE (Institute for Experimental Software Engineering) radicado en Alemania. Está organizada en 3 conjuntos de elementos como muestra el Anexo 1 (36) (37).

- Conjunto de etapas que comprenden el ciclo de vida: inicio, desarrollo, uso y evolución.
- Conjunto de componentes técnicos o herramientas que dan soporte a cada una de estas etapas. Aparecen en el Anexo 1, con el mismo color que la fase del proceso en la que se emplean.
- Componentes de soporte, que permiten evaluar la calidad de la aplicación del método, en cierto entorno.

El modelo está correctamente documentado, usa notaciones como (Unified Modeling Language, UML), sigue el enfoque orientado a componentes, utiliza la disciplina de la Arquitectura de Software, cuenta con un expediente medianamente completo, si embargo no especifica un proceso de gestión de repositorios (ver Anexo 8).

Metodología KobrA: aglutina varias tecnologías avanzadas de ingeniería de software entre las que se incluyen el desarrollo de LPS, el desarrollo de software basado en componentes, el diseño de fameworks[1] y arquitecturas de software, el uso de métricas de calidad o el modelado de procesos. El proyecto KobrA es financiado por el Gobierno Alemán y se lleva a cabo por un consorcio de cuatro organizaciones: Softlab GmbH, Munich, Psipenta GmbH, Berlín, GMDFIRST, Berlín y el IESE Fraunhofer. El proceso se divide en las siguientes etapas:

1. Ingeniería de Dominio: permite definir un marco genérico de desarrollo en el que se definen todas las posibles variantes dentro de una familia de productos (incluyendo información sobre sus características comunes y diferentes). Los tres subprocesos que comprenden la ingeniería de dominio son: definición del contexto, especificación de componentes y definición de componentes.
2. Ingeniería de aplicaciones: permite instanciar el marco de desarrollo para crear distintos productos, cada uno de ellos diseñados para satisfacer las necesidades específicas de los distintos clientes. Este proceso de divide en dos etapas fundamentales: instanciación del contexto de la LPS e instanciación del framework de la LPS (38) (36).

Su asimilación y uso no es trivial, se basa en estándares y realiza trabajo con el repositorio se pero pudiera explotarse más en este sentido, presenta un expediente medianamente completo, su fortaleza fundamental está en el manejo de la disciplina Arquitectura de Software (Anexo 8).

Proceso de desarrollo PRAISE (Product-line Realisation and Assessment in Industrial Settings): es un proceso definido en el ESI (European Software Institute). PRAISE propone un esquema parecido al ciclo de vida en cascada, añadiéndole una fase de análisis de dominio, cuyo resultado es la

[1] Marco de trabajo

elaboración de una arquitectura de referencia, como se muestra en el Anexo 2. La implementación tiene como resultado la producción de componentes reutilizables (39) (40).

La Ingeniería de Familias de Aplicaciones produce los activos que el proceso de Ingeniería de Aplicaciones reutilizará, involucrando las siguientes actividades: Análisis del Dominio, Diseño del Dominio e Implementación del Dominio.

Las actividades que comprende la Ingeniería de Aplicación son: Análisis de los requisitos de la aplicación, Diseño del producto y Codificación del producto (40).

Se definen correctamente la Ingeniería de Dominios y de Aplicaciones lo que hace de simple asimilación, se propone una Arquitectura de Referencia, pero no se definen actividades de gestión de repositorios, ni se maneja la gestión de costos y riegos. Los artefactos son pobres, así como el uso de estándares (ver Anexo 8)

Modelo de referencia CAFE (From Concepts to Application in System-Family Engineering): es la continuación del proyecto ESAPS (Engineering Software Architectures, Processes and Platforms for System-Families). Uno de sus resultados es el Entorno de Referencia de Familias de Sistemas que combina cuatro modelos diferentes a su vez:

1. Ciclo de vida de una organización, que trabaja en el campo de las Tecnologías de la Información: ofrece una visión global de alto nivel de la organización, identificando dos actividades principales, el uso de la familia de productos (desarrollo, despliegue y mantenimiento) y el pre-uso de la familia de productos (planificación, análisis económico y delimitación).
2. Entorno de Procesos ISO/IEC 15.504, que clasifica todos los procesos de una entidad en cinco grupos: Cliente- Suministrador, Ingeniería, Organización, Gestión y Soporte.
3. Modelo de referencia CAFE (CAFE CRM), que define seis grandes paquetes básicos de trabajo en una línea de productos.
4. Modelo de activos CAFE (CAFE ARM), clasifica los diferentes activos de la línea de productos según el modelo de referencia anterior.

El Anexo 3 muestra el modelo general de referencia del proyecto CAFE, donde se evidencian las actividades de prueba, tanto para la Ingeniería de Dominios como de Aplicaciones.

Es un modelo bastante integral y bien especificado, con fortaleza en los procesos de gestión, propone un conjunto de artefactos que dan soporte al modelo, sin embargo no define actividades que incluyan la gestión del repositorio de los activos generados y los elementos de arquitectura manejados son muy pobres (ver Anexo 8).

Modelo TWIN: modelo de procesos de dos fases o modelo TWIN Life Cycle, es empleado en el desarrollo de software basado en componentes. La esencia de este modelo se basa en los procesos de desarrollo de activos reutilizables y de desarrollo de aplicaciones basadas en la reutilización:

1. Desarrollo de software para la reutilización: donde el propósito es producir componentes de software reutilizables (este proceso se denomina Ingeniería de Dominio).
2. Desarrollo de software con reutilización: cuyo propósito es el desarrollar software reutilizando componentes existentes (este proceso recibe el nombre de Ingeniería de Aplicación).

Tanto la Ingeniería de Dominios como la Ingeniería de Aplicación proponen métodos que implican la reutilización, ya sea de activos reutilizables o de componentes reutilizables. Los procesos del modelo TWIN se representan en el Anexo 4 (31) (39).

Identifica claramente los procesos de Ingeniería de Domino y de Aplicaciones, trabaja la disciplina de la Arquitectura y deja claro los mecanismo de reutilización de activos y componentes. Sus dificultades están en los procesos de gestión y la calidad de los estefactos (ver Anexo 8)

1.2.2 Modelos americanos

Modelo WATCH: propuesto en la Universidad de Los Andes (Venezuela) para el desarrollo de aplicaciones empresariales. Consta de dos componentes metodológicos:

1. Ingeniería de Dominio: expresado en el método WATCH-Component encaminado al desarrollo de componentes de software reutilizables. El grupo de procesos fundamentales se muestran en el Anexo 5 (31).
2. Ingeniería de Aplicaciones: expresado en el método WATCH-Application encaminado al el desarrollo de aplicaciones empresariales. El grupo de procesos fundamentales se muestran en el Anexo 6 (31).

Utiliza UML como lenguaje de modelado, emplea el estándar IEEE 1074 para crear la estructura de procesos del modelo, es simple y adaptable. Plantea la documentación del proceso. Define un repositorio donde gestionar productos y activos reutilizables de la línea. El trabajo con la Arquitectura es superficial. (ver Anexo 8)

Modelo SPLEP (Evolutionary Software Product Line Engineering Process): es un proceso de desarrollo de software que potencializa el desarrollo y mantenimiento de software. Es un proceso altamente iterativo, donde los productos evolucionan continuamente, lo que hace que los sistemas se puedan adaptar fácilmente a los requisitos cambiantes del entorno de negocio. Además, como los nuevos productos son a menudo evoluciones de los ya existentes o están conformados por partes reutilizables de estos, el modelo ESPLEP permite el desarrollo de las familias de productos de software tal como se muestra en el Anexo 7 (31). Las fases de Ingeniería de la Línea de Productos de Software del método SPLEP son:

- Modelado de requisitos de la línea.
- Modelado de análisis de la línea.
- Modelados de diseño de la línea.
- Implementación incremental de los componentes.

- Pruebas de la línea de productos.

Todos los componentes producidos son entregados al repositorio de la LPS. Por otro lado la fase de la Ingeniería de Aplicaciones del método SPLEP se encarga de especificar cómo estas actividades van encaminadas a la personalización de los activos guardados en el repositorio a clientes concretos.

Plantea como elemento fundamental el repositorio de activos, usa UML y se documenta el proceso. La disciplina de la Arquitectura de Software se maneja superficialmente. Los procesos de gestión son pobres (ver Anexo 8).

Modelo propuesto por el SEI: el Modelo de procesos de LPS desarrollado por el SEI que se muestra en la Ilustración 4 se centra en tres elementos fundamentales:

- Desarrollo de Activos básicos, que conforman en la Ingeniería de Dominio y tiene el objetivo de establecer la capacidad de producción para los productos, mediante el desarrollo de activos de Software. El desarrollo de activos básicos define el alcance de la línea, el plan de producción y los activos en sí.
- Desarrollo de Productos, que hace alusión a la Ingeniería de de Aplicaciones y tiene como objetivo elaborar los productos de la línea a partir del ensamblaje de activos básicos siguiendo el plan de producción. El resultado de este proceso son los productos acabados.
- Gestión de la Línea de Productos cuyo objetivo es proporcionar los recursos, coordinar y supervisar el desarrollo de activos y productos. Esta gestión se divide en: gestión técnica (orientada a los grupos que desarrollan activos y productos) y gestión organizacional (orientada a los aspectos organizacionales como son: estructura, relaciones, recursos, financiamiento, etc.) (5).

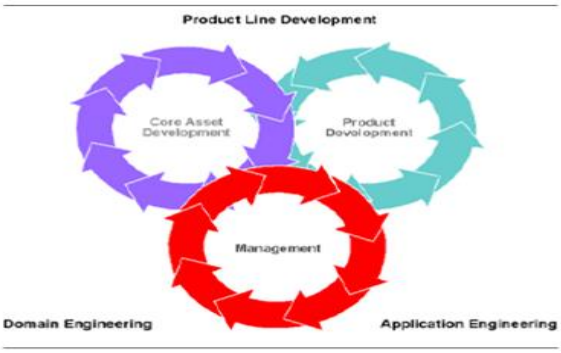


Ilustración 4 Modelo de procesos de LPS desarrollado en el Software Engineering Institute (SEI) (5)

La Tabla 3 resume las áreas prácticas del modelo propuesto por el SEI, donde el área de Ingeniería de Software incluye las actividades relacionadas con la Ingeniería de Dominio y de Aplicaciones, mientras la Gestión de la Línea de Productos se materializa en las áreas de Gestión Técnica y Organizacional.

Tabla 3 Áreas Prácticas del modelo LPS propuesto por el SEI (6)

Ingeniería de Software	Gestión Técnica	Gestión Organizacional
Definición de la arquitectura	Gestión de la configuración	Construcción de casos de uso del negocio
Evaluación de la arquitectura	Análisis de factibilidad	Gestión de clientes
Desarrollo de componentes	Seguimiento y control	Gestión de adquisición
Minería de activos existentes	Definición del proceso	Gestión de costos
Ingeniería de requisitos	Gestión del alcance	Implantación y soporte
Integración de sistemas	Planificación técnica	Análisis de mercado
Pruebas	Gestión de riesgos técnicos	Operaciones
Comprensión de Dominios relacionados	Herramientas de soporte	Planificación de la organización
Reutilización de software de terceros		Gestión de riesgos de la organización
		Estructuración de la organización
		Observatorio tecnológico
		Entrenamiento

Es el modelo tomado como referencia a nivel internacional, es un proceso completo y bien documentado, usa estándares, maneja los conceptos de arquitectura de software y repositorio. Su principal dificultad radica en la complejidad y alcance (ver Anexo 8).

Modelo LPS desarrollo GESPRO[2]: está basado en los principios de las LPS, la arquitectura de empresas y la mejora continua. Incluye un guía para su implantación, donde se detallan cada uno de los pasos a seguir. Desarrollado en Cuba, en la UCI (Universidad de las Ciencias Informáticas), por el Ing. Henrik Pestano Pino como trabajo para optar por el título de máster en ciencias. Toma como punto de partida las experiencias obtenidas en la organización del Centro de Tecnologías de Almacenamiento y Análisis de Datos y de la Dirección Técnica de Producción de la propia institución. Las características fundamentales son:

- Modelo centrado en arquitectura de empresas: se presenta dividido en cinco vistas (Negocio, Datos, Aplicaciones, Tecnología y Recursos Humanos) para facilitar la flexibilidad en los cambios del negocio y los sistemas, de manera coherente con la implantación y el desarrollo de LPS y soportando además la mejora continua.
- Modelo centrado en la mejora continua: la mejora continua no es un paso sino la aproximación al modelo ideal por iteraciones, donde se repiten cada uno de los pasos partiendo de lo aprendido en la iteración anterior. Estos ciclos se repiten de forma permanente durante el tiempo de vida de la organización y son la esencia de la mejora continua.
- Modelo centrado en líneas de productos de software: plantea cinco elementos fundamentales que se revisan en cada una de las vistas anteriormente definidas (Dominio, Familia de productos,

[2] Herramienta de Gestión de proyectos

Modo de producción, Arquitectura, Activos de software). (39)
Constituye una guía para la implantación de una LPS. Usa los conceptos de arquitectura y repositorio. Sus debilidades radican en la claridad de los artefactos que documentan el proceso y el uso de estándares (ver Anexo 8).

1.2.3 Análisis del estudio de los modelos LPS

El trabajo realiza una comparación de los modelos de LPS estudiados, respecto a un grupo de indicadores metodológicos que se creen imprescindibles en un modelo LPS para el contexto nacional al cual se dirige la investigación. A cada indicador se le da una evaluación del uno al cinco, obteniendo mejor puntuación el que presente mayor acercamiento a los indicadores.

Facilidad de asimilación y uso: se refiere a la calidad que posee la documentación del modelo para su explicación. El tiempo que se necesita para entenderse e implantarse, entre otros.

Estándares: se refiere al uso de estándares que el modelo propone o utiliza (lenguajes formales, estándares de repositorio, nomenclaturas de elementos, etc.).

Repositorio: se refiere a la presencia en el modelo de repositorios, así como el nivel de calidad de manejo del mismo.

Arquitectura: se refiere al nivel de definición de procesos arquitectónicos en el modelo.

Expediente: se refiere a la presencia de planillas y artefactos y su organización en portafolios, de acuerdo a las fases o procesos del modelo.

Según lo arrojado por el análisis anterior (ver Anexo 8), el modelo que pondera con mayor nivel de acercamiento a los indicadores planteados es el propuesto por el SEI. Este constituye el escogido como base para la presente investigación.

1.3 Metodologías de desarrollo de software.

Metodología RUP (Rational Unified Process): es una metodología de desarrollo de software robusta. Tuvo una última versión en el año 2003. Sus principales autores fueron Ivar Jacobson, Grady Booch y James Rumbaugh. Tiene una gran integración con UML (Lenguaje Unificado de Modelado). Cuenta con tres características fundamentales: dirigida por casos de uso, centrada en la arquitectura, iterativa e incremental. Posee cuatro fases fundamentales: inicio, elaboración, construcción, transición y nueve flujos de trabajos: modelado del negocio, requisitos, análisis y diseño, implementación, pruebas, despliegue, gestión de la configuración, gestión de proyectos y entorno, como se muestra en el Anexo 9 (41) (42) (43).

Metodología XP (Extremer Programming): es una metodología de desarrollo de software ágil creada por Kent Beck y usada en proyectos de corto equipo y corto plazo de entrega. Esta metodología presenta un grupo de conceptos fundamentales entre los que se encuentran:

planificación incremental, testing, programación en parejas, refactorización, diseño simple, propiedad colectiva del código, integración continua, cliente en el equipo, entregas pequeñas, semanas de 40 horas, estándares de codificación y uso de metáforas. El proceso fundamental de la metodología XP se muestra en el Anexo 10. Las fases de fundamentales de XP son: planificación, diseño, desarrollo y pruebas. Además cada fase presenta un grupo de elementos como se muestra en el Anexo 11 (44) (9) (42) (43) (45).

Metodología CRYSTAL: se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Propuesta por Alistair Cockburn a principio de los 90. La idea es poder armar distintas metodologías para distintos tipos de proyectos. Cada proyecto y organización usará un grupo de elementos genéticos para generar su propia metodología. El nombre Crystal se deriva de la caracterización de los proyectos según las dimensiones tamaño y complejidad. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (46) (47) (42) (43).

Metrología MSF (Microsoft Solutions Framework): es un conjunto de principios, modelos, disciplinas, conceptos, lineamientos y prácticas probadas elaborado por Microsoft. Controla la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. Presenta un grupo de modelos fundamentales entre los que figuran: modelo de arquitectura del proyecto, modelo de equipo, modelo de procesos, modelo de gestión del riesgo, modelo de diseño del proceso y modelo de aplicación. Propone además cinco fase fundamentales: visión y alcance, planificación, desarrollo, estabilización e implementación, como se muestra en el Anexo 12 (42) (48) (43) (49).

Metodología DSDM (Dynamic Systems Development Method): originada en los trabajos de Jennifer Stapleton en el año 1994. Además de un método ágil proporciona un framework completo de controles para RAD y lineamientos para su uso. Puede complementar las metodologías XP, RUP o MSF o combinaciones de todas ellas. Consiste en tres fases: fase del pre-proyecto, fase del ciclo de vida del proyecto y fase del post-proyecto. La fase del ciclo de vida del proyecto se subdivide en cinco etapas: estudio de viabilidad, estudio de la empresa, iteración del modelo funcional, diseño e iteración de la estructura e implementación. Existen además nueve principios subyacentes a la metodología, consistentes en cuatro fundamentos y cinco puntos de partida para la estructura del método. Estos principios forman los pilares del desarrollo:

- Involucrar al cliente es la clave.

- El equipo del proyecto debe tener el poder.
- Entrega frecuente de productos.
- Entregar un sistema que satisface las actuales necesidades de negocio.
- El desarrollo es iterativo e incremental.
- Todos los cambios durante el desarrollo son reversibles.
- Alcance de alto nivel y los requerimientos deberían ser base-lined.
- Las pruebas son realizadas durante todo el ciclo vital del proyecto.
- La comunicación y cooperación entre todas las partes interesadas en el proyecto (42) (43).

Metodología SCRUM: desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle Scrum. Constituye una metodología ágil, iterativa e incremental de desarrollo de software. Como se observa en el Anexo 13, esta metodología estructura el desarrollo en ciclos de trabajo llamados sprints que tienen una duración fija y pequeña. Cada sprint aporta el incremento de una parte del producto. Al comienzo de cada sprint, un equipo multi-funcional selecciona los elementos (requisitos del cliente) de una lista priorizada, se planifican los sprints y se comienza el trabajo. Todos los días el equipo se reúne brevemente para informar del progreso. Al final del sprint, el equipo revisa el resultado con los interesados para mostrarle lo que han construido y además se retroalimentan de las lecciones aprendidas (50) (51). En SCRUM se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al cliente. Por ello, SCRUM está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados tempranos, los requisitos son cambiantes o poco definidos y la innovación, la competitividad y la productividad son fundamentales (52).SCRUM plantea la realización de tres reuniones: planificación, seguimiento y revisión del sprint.

***Planificación del sprint*:** en esta reunión se define el Product Backlog, el cual consiste en una lista priorizada de requisitos del sistema y es un documento vivo, que puede ser continuamente actualizado. En cada iteración el Product Backlog es revisado por el equipo.

***Seguimiento del sprint*:** se llevan a cabo breves reuniones diarias, para ver el avance de las tareas y el trabajo que está previsto para la jornada.

***Revisión del sprint*:** una vez finalizado el sprint se realiza un análisis y revisión del incremento generado, se presentan los resultados finales y se recomienda siempre tener preparado un demo.

SCRUM plantea tres tipos de roles: el propietario del producto, el scrum máster y el equipo.

***Propietario del producto*:** es la persona conocedora del entorno de negocio del cliente y de la visión del producto, representa a todos los interesados en el producto final y es el responsable del Product Backlog de obtener el resultado de mayor valor posible para los usuarios o clientes, de la financiación necesaria para el proyecto y de decidir cómo debe ser el resultado final del proyecto.

***Scrum máster*:** es el encargado de garantizar el funcionamiento de los procesos y de la metodología. Debe interactuar tanto con el equipo como con el cliente y con los gestores. Es

responsable de garantizar que el proceso sea entendido y seguido, asegurando que el equipo se adhiera a los valores, prácticas y normas de SCRUM.

***Equipo de desarrollo*:** es el equipo del proyecto y tiene la autoridad para decidir en las acciones necesarias y para auto-organizarse. Los miembros del equipo deben tener todas las habilidades necesarias para crear un incremento de trabajo.

SCRUM no requiere de ninguna práctica concreta para el desarrollo del software, sin embargo sí dispone de prácticas y herramientas para la gestión de las diferentes fases. Los dos conceptos fundamentales respecto a este enfoque son:

***Product Backlog*:** define los requisitos del sistema o el trabajo a realizar a lo largo del proyecto. Está compuesto por una lista de requisitos de negocio y técnicos, actualizados y priorizados.

***Sprint Backlog*:** es una lista de trabajos que el equipo se compromete a realizar para generar el incremento previsto. Las tareas están asignadas a personas y tienen estimados el tiempo y los recursos.

1.3.1 Análisis de criterios según Boehm y Turner

El análisis de territorios planteado por Boehm y Turner permite conocer las condiciones bajo las cuales cada metodología (ágil o robusta) tiene más probabilidad de éxito. Parte de cinco principios fundamentales: tamaño, criticidad, dinamismo, personal y cultura. Los factores que permiten clasificar, tomando las restricciones o características impuestas por el problema a resolver y los medios disponibles para hacerlo consideran al: proyecto, producto, organizaciones y personas que ejecutan, o están relacionas al contexto de negocio. Según la matriz de propuesta por Boehm y Turner para la definición del tipo de metodologías a emplear los esquemas que se acerquen más al centro (representados en color azul) se complementan con metodologías ágiles, los esquemas intermedios (representados en color verde) se complementan con metodologías mixtas y los más alejados (representados en color naranja) se complementan con metodologías orientadas al plan, como se muestra en la Ilustración 5 (53) (54).

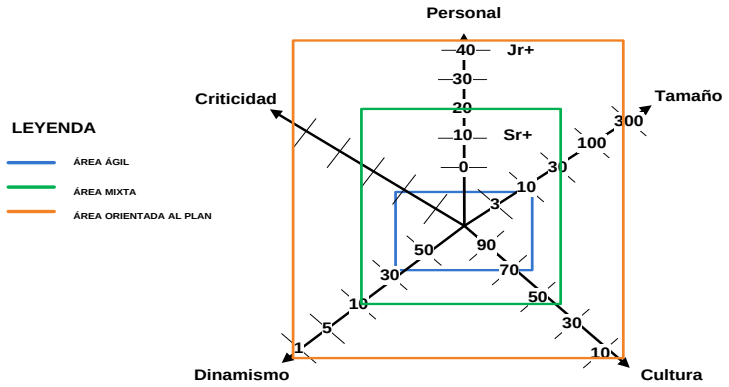


Ilustración 5 Análisis de territorios según Boehm y Turner

El (Anexo 14) muestra una comparación entre las metodologías ágiles y tradicionales con respecto a un grupo de indicadores (equipo, requisitos, arquitectura, fases, etc), pudiéndose concluir que las metodologías tradicionales se caracterizan por ser métodos pesados y formales que requieren una basta documentación, realizan planificaciones del desarrollo del producto en etapas tempranas. Se realiza además, gran énfasis en la definición del proceso: roles, actividades y artefactos. Por otro lado las metodologías ágiles se ajustan a los principios del manifiesto ágil:

- El software que funciona por encima de la documentación exhaustiva.
- Los individuos y su interacción por encima de los procesos y las herramientas.
- La colaboración con el cliente por encima de la negociación contractual.
- La respuesta al cambio por encima del seguimiento de un plan (4).

Todo esto apunta a la simplicidad del proceso de desarrollo, la centralización en los recursos humanos como eslabón fundamental de la cadena, incluyendo al cliente como parte del proceso de desarrollo, la escasa documentación, los requisitos, entre otros. El (Anexo 15) muestra una comparación entre algunas metodologías ágiles que estudia la presente investigación (SCRUM, XP, DSDM, CRYSTAL), constituyendo SCRUM una de las metodologías ágiles más usadas a nivel internacional, que promueve la autogestión del equipo y muy fácil de entender e implementar.

Conclusiones parciales

- Entre los modelos de desarrollo de software estudiados más novedosos y productivos se encuentra el modelo basado en LPS. Todos los modelos de LPS estudiados plantean las fases de Ingeniería de Dominios para el desarrollo de los elementos comunes (activos básico) de la línea y la fase de Ingeniería de Aplicaciones para el desarrollo de aplicaciones. De los diferentes modelos de LPS estudiados, el propuesto por el SEI presenta una alta tendencia a la reutilización y la semiautomatización del desarrollo basado en estándares; constituyendo un modelo ventajoso para tomar como base teórica en la presente investigación.
- Existen dos enfoques fundamentales en las metodologías de desarrollo de software: ágil y robusto, siendo estos enfoques un punto de análisis para determinar cuál de ellos se adapta de una mejor forma a las empresas SICS y DESOFT. De las metodologías ágiles estudiadas, SCRUM promete ser factible para utilizar en estos referidos entornos, por sus características y buenas prácticas comprobadas, su nivel de formalización y la alta productividad que demuestra. Además de la facilidad de implementar, que no necesita altos porcientos de especialización de la gran masa del equipo de desarrollo.

CAPÍTULO 2 DESARROLLO DE LA SOLUCIÓN

Introducción

En el presente capítulo se explican los conceptos fundamentales del modelo LPS propuesto. Se explica cada una de las fases de este, detallando flujo de actividades y artefactos fundamentales. El modelo centra sus postulados en la Ingeniería de Dominios (disciplina que se encarga de la realización de los elementos comunes de la LPS), Ingeniería de aplicaciones (disciplina encargada de desarrollar las aplicaciones a clientes concretos), Administración (disciplina encargada de la gestión y gerencia de las LPS) y Arquitectura como disciplina que se le adhiere al modelo, con el objetivo de lograr una abstracción arquitectónica de la solución antes de ejecutar su producción. Las fases fundamentales que el modelo plantea son: Concepción, Conceptualización Arquitectónica, Ingeniería de Dominios, Ingeniería de Aplicaciones y Empaquetado y Despliegue, todas ellas gestionadas por la metodología de desarrollo SCRUM y los principales postulados de gestión de proyectos.

2.1 Conceptos fundamentales del Modelo

La presente investigación toma de base de referencia el modelo propuesto por el SEI, planteando el desarrollo de activos básicos, desarrollo de aplicaciones y gestión de la LPS en correspondencia con este. Adicionalmente se incorporan la disciplina de Arquitectura, quien centra el proceso de desarrollo y como mecanismo metodológico SCRUM. Para estar en correspondencia con estos conceptos, se plantea que la Arquitectura provea dos abstracciones arquitectónicas del producto, una referida al dominio y la otra a las aplicaciones, como muestra la Ilustración 6.

Independientemente del modelo de desarrollo arquitectónico que se use, la disciplina de Arquitectura debe proporcionar la abstracción de alto nivel de la solución corporativa del producto, vista desde distintas perspectivas tales como: procesos, sistema, datos integración, tecnología, información e infraestructura. La **Arquitectura de Dominio** se encarga de definir estas vistas abstractas del producto para los elementos comunes que están en correspondencia con el segmento de mercado en que se centra la LPS. Por otro lado la **Arquitectura de Aplicaciones** representa una instancia de la Arquitectura de Dominios, ajustada a los entornos de clientes concretos. La Arquitectura de Aplicaciones está representada por la Arquitectura de Dominio más la variabilidad que demanda la adecuación al cliente.

El modelo incluye el concepto **Desarrollo del Dominio**. Encargado de la construcción de los activos básicos que conforman los elementos comunes de la familia de productos definidos en la Arquitectura de Dominios. Estos activos pueden ser tecnológicos o temáticos, en dependencia de si responden elementos tecnológicos o a componentes que automatizan procesos de negocio.

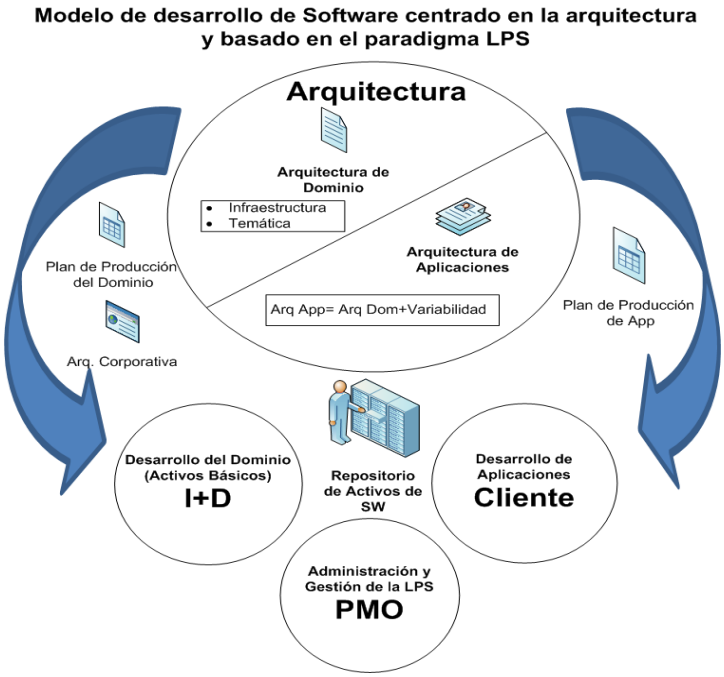


Ilustración 6 Modelo de Desarrollo de Software centrado en la arquitectura y basado en el paradigma LPS

El otro elemento del modelo **Construcción de Aplicaciones** Instancia los componentes del dominio con el objetivo de desarrollar los productos enfocados a clientes El mismo utiliza la infraestructura creada en el dominio para construir un representante de la familia de productos, según el cliente que lo demanden. Se parte de requisitos de clientes que deben estar alineados con las características modeladas en la arquitectura de dominio, aunque pueden existir variaciones que sea necesario desarrollar para este. El estudio de estos requisitos produce un análisis detallado de la variabilidad de su construcción, así como la cantidad de elementos que se pueden reutilizar, proceso que centra la arquitectura de aplicaciones. El producto se configura a partir de los activos básicos, ensamblando los elementos comunes y variables para producir el resultado final (35)

El **Repositorio de Activos de Software** es el elemento del modelo encargado de centralizar los activos de software que se desarrollan en la Construcción de Dominios y Aplicaciones. Funciona como una base de datos especializada que además de almacenar los activos, también facilita la recuperación y el mantenimiento de estos. Su objetivo fundamental es asegurar la disponibilidad de activos para apoyar el desarrollo de productos El Repositorio gestiona y controla además la documentación asociada a los activos. (31)

Por último, otro elemento importante del modelo lo constituye el área de **Gestión de la LPS**. Se encarga de asignar recursos, coordinar y supervisar el desarrollo de activos y productos. Esta gestión se realiza tanto en el nivel técnico (gestión de proyecto), como en el nivel organizativo (estructura organizativa adecuada a los objetivos propuestos por la LPS) (35). La gestión técnica es

enfocada en algunos de los procesos señalados por el PMBOOK[3], como son: la gestión de planificación, la gestión de alcance y tiempo, la gestión de riesgos, el seguimiento y control, la gestión de costo y la gestión de calidad, entre otros elementos metodológicos (55). El nivel organizacional está relacionado con la composición de la empresa y las actividades que en ella deben implantarse para asegurar el aprovechamiento eficaz y eficiente del paradigma LPS.

A partir de los conceptos que plantea el modelo, se proponen cinco elementos fundamentales que deben ser contemplados:

- Dominio: se refiere al área de aplicación de productos de software del entorno de negocio donde se despliega la línea de productos y su posible integración a ecosistemas de software.
- Familia de productos: define el conjunto de software asociados a un dominio determinado. Los miembros de la familia comparten aspectos comunes tales como:
  - Diseño arquitectónico común.
  - Conjunto componentes reutilizables.
  - Capacidades y servicios comunes.
  - Tecnologías comunes.
- Modo de producción: se refiere a la forma de organización del proceso productivo
- Arquitectura: incluye toda la información referente a la línea base de la arquitectura de las soluciones.
- Activos de software: son todos los artefactos y componentes que se producen con el propósito de ser reutilizados múltiples veces en el desarrollo de aplicaciones. (39)

2.2 Modelo de general de procesos

El Modelo general presenta como características fundamentales que es centrado en la arquitectura y basado en el paradigma LPS. Consta con cinco fases fundamentales ver Ilustración 7.

- Concepción: encargada de estudiar y conceptualizar la idea del surgimiento de nuevos proyectos provenientes del análisis de entornos de negocio o de clientes concretos.
- Conceptualización Arquitectónica: proporciona la Arquitectura de dominios de la solución expresada en vistas (Proceso, Sistema, Datos, Integración, Infraestructura, Información y Tecnología). Desagrega los elementos fundamentales en dos instancias arquitectónicas: Arquitectura de Dominios y Arquitectura de Aplicaciones.
- Ingeniería de Dominios: desarrolla los activos básicos que constituyen los elementos comunes de la familia de productos que desagrega la Arquitectura de Dominio de cada LPS. Estos pueden ser componentes temáticos especializados o tecnológicos. Una vez construidos se envían al

[3] Guía de los Fundamentos de la Dirección de Proyectos del PMI (Project Management Institute)

repositorio de Activos de Software, donde son guardados, versionados y mantenidos para su posterior reutilización en la fase de Construcción de Aplicaciones.

- Ingeniería de Aplicaciones: encargada del desarrollo de cara al cliente, se basa en la reutilización de los activos de software núcleos desarrollados en el dominio.
- Empaquetado y Despliegue: encargada del empaquetado del producto. Como resultado se logra agrupar todos los complementos que los activos o productos necesarios para el despliegue, puesta en marcha y mantenimiento. Garantiza la distribución del producto y su correcta implementación, así como la retroalimentación de la experiencia adquirida.

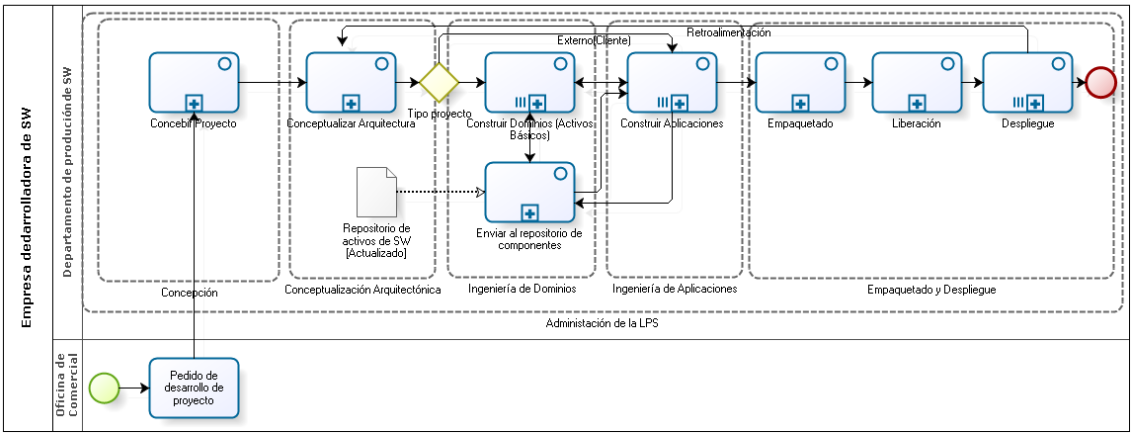


Ilustración 7 Modelo general de procesos de desarrollo se software centrado en al arquitectura y basado en el modelo LPS

El desarrollo de un producto puede responder tanto al trámite realizado por un cliente como al desarrollo de arquitecturas y tecnologías genéricas orientado a un segmento de mercado, que ofrezca potencialidades de comercialización. De esta manera se inicia la primera fase del modelo (Concepción) y con ella el subproceso Concebir proyecto, seguido del subproceso Conceptualizar arquitectura de la fase de Conceptualización arquitectónica. En dependencia del tipo de proyecto requerido (enfocado al desarrollo de un dominio o personalización a clientes) se determina si el flujo fluye al subproceso Construir Dominio de la fase Ingeniería de Dominio encargada del desarrollo de bases tecnológicas y/o áreas temáticas, o al subproceso Construir Aplicaciones de la fase de Ingeniería de aplicaciones. Al finalizar cualquiera de estos se ejecuta el Subproceso Enviar al repositorio de Componentes. Al terminar las fases de Ingeniería de Aplicaciones, se procede a la fase de Empaquetado y Despliegue que ejecuta los subprocesos de Empaquetado, Liberación y Despliegue, los que permiten responder al cliente y terminar el proceso general retroalimentando con las experiencias adquiridas a las fases que le antecedieron.

Es importante aclarar que se pueden conceptualizar y definir arquitecturas para cada entorno de negocio que se determine abarcar. Se deben construir tantos dominios como LPS existan, tantas aplicaciones como clientes concretos demanden y se empaqueta tantas veces se entreguen productos a clientes concretos.

Durante todo el proceso de desarrollo se realiza la Administración de la LPS, con el objetivo de llevar a cabo un conjunto de actividades de gestión tanto técnica como organizacional, durante toda la ejecución del proceso productivo. Entre los principales elementos tomados en cuenta en esta área destacan, riesgos, planificación, alcance, calidad, costos y seguimiento y control. Se propone usar como metodología de desarrollo SCRUM.

2.3 Fases del modelo

2. 3.1 Fase de Concepción

Subproceso Concebir Proyecto

A partir de la solicitud de un pedido de desarrollo realizada por el área comercial, se puede decir que existen dos tipos de proyectos:

Proyectos externos: Constituyen desarrollos de personalización orientados a empresas del segmento de mercado, para los que se realizan adaptaciones a la medida de los arquetipos arquitectónicos existentes Dependiéndose del grado de cubrimiento existente entre los arquetipos desarrollados y el entorno de negocio del cliente, se proyectan soluciones personalizadas que permiten un cubrimiento de las necesidades demandadas. Es significativo resaltar que estas constituyen la fuente de ingreso de la organización productora, cuyos indicadores de utilidad ascienden en la misma medida en que se logren reutilizar activos de software durante las personalizaciones.

Proyectos internos: constituyen demandas de desarrollo basado en el principio I+D+I enfocados a crear base tecnológica especializada en un entorno organizacional específico. Como resultados de estos proyectos se desprenden arquetipos arquitectónicos especializados. Mediante un proceso de composición orientado a componentes, estos arquetipos son construidos creándose una infraestructura temática adaptable y configurable a los clientes concretos de estos entornos organizacionales.

El Anexo 16, muestra el flujo de actividades del subproceso Concebir Proyecto. Este presenta una bifurcación inicial que responde a la interrogante de: ¿El proyecto que se pretende desarrollar es de tipo externo o interno? En el caso de que el proyecto sea externo se procede al siguiente flujo de actividades:

- Encuentro inicial con el cliente: Se realiza un primer contacto entre el equipo de desarrollo y el cliente, donde este último expone sus necesidades y objetivos. El equipo del proyecto realiza preguntas sobre el proceso de negocio a informatizar, los requerimientos que el cliente delimita y las restricciones que los procesos y características de la empresa imponen.

Aplicar arquetipo arquitectónico: se procede a aplicar la verificación y chequeo del nivel de correspondencia entre la arquitectura de dominios y el entorno del cliente. Se identifican las

variaciones del dominio y se proyecta un primer acercamiento a la Arquitectura de Aplicaciones, ambas acciones sirven como entrada a la fase siguiente, que se encarga de abstraer arquitectónicamente la personalización del producto al cliente concreto. Algunas interrogantes claves que permiten el macheo de la arquitectura tipo a las necesidades del cliente concreto son:

- ¿Los procesos que desea informatizar la empresa cliente son cubiertos por la arquitectura de dominio?
- ¿Las reglas de negocio y estándares requeridos se rigen por las regulaciones jurídicas y estándares implementados por la arquitectura de dominio?
- ¿La infraestructura tecnológica presente en la empresa cliente soporta los requerimientos de infraestructura señalados por la arquitectura de dominio?
- ¿Las restricciones y requerimientos de seguridad demandada por la empresa están contempladas en las especificaciones de la arquitectura de dominio?

En caso de que exista correspondencia entre el arquetipo arquitectónico y el entorno de negocio cliente, y el esfuerzo de personalización sea económica y técnicamente factible; se procede a la formalización del proyecto.

- Confección del contrato: se describen los acuerdos de voluntades entre las partes fijándose los derechos y obligaciones relativos. Se establece acápites sobre las responsabilidades de ambas partes en el proyecto y el producto. Es función elemental del contrato originar efectos jurídicos.
- Confección del proyecto técnico: describe las tecnologías a utilizar, sus potencialidades, debilidades, características y requerimientos. Se describe el entono de negocio al que se le pretende plantear la solución de software. Se formula la propuesta abordando sus objetivos, alcance y riegos. Se comenta la organización del proyecto. Se define la arquitectura de la solución. Se establecen temas de calidad, despliegue, capacitación, transferencia de conocimientos y soporte técnico del software en caso que el proyecto lo demande, productos entregables, entre otros.

Posterior a la confección del proyecto técnico, se emite notificación a la fase de Concepción de la Arquitectura para llevar a acabo la personalización de la Arquitectura de Dominio al entorno de negocio específico del cliente. Lo que con otras palabras sería definir la Arquitectura de Aplicaciones.

Por otro lado si no se encuentra correspondencia entre el arquetipo arquitectónico y el entorno de negocio cliente, sucede el siguiente flujo:

- Comunicar al cliente: se le explican las razones al cliente que hacen que la empresa productora no enfrente el desarrollo de la solución.
- Archivar razones de denegación: se guarda la documentación que avala las razones por las que se denegó el desarrollo de la solución al cliente dando fin al proceso.

En el caso de que el proyecto sea de tipo interno, se procede al siguiente flujo de actividades:

- Estudio del estado del arte del entorno de negocio: se estudia, valora y evalúa el entorno de negocio, encaminado a profundizar en los siguientes objetivos:
  - Conocer el entorno de negocio, los procesos claves, estratégicos, soporte y capacidades comerciales.
  - Identificar características tecnológicas, legales y funcionales que deberá satisfacer el resultado del producto, y por consiguiente los lineamientos jurídicos de las tecnologías y metodologías a utilizar en el desarrollo de la solución.
  - Identificar productos competidores, sus hitos funcionales y tecnológicos más importantes que serán referencias en la definición del alcance y prestaciones de los productos que montará la posible solución.
  - Realizar un estudio de viabilidad y factibilidad técnica para la posible solución enmarcada en el índice de crecimiento que se prediga del análisis económico, capacidad de producción, y duración del ciclo de vida del proceso productivo.
  - Identificar a partir de las características del entorno de negocio encontradas, los componentes o soportes tecnológicos que deberán ser contratados e instanciados para la solución (componentes tecnológicos, hardware o legales) o para la posible ejecución del proceso productivo.
  - Identificar las familias de producto, acorde a las líneas temáticas asociadas a los procesos globales identificados en los pasos predecesores.

A partir del estudio del estado del arte realizado y habiendo aplicado un estudio de factibilidad se puede conocer si es conveniente realizar el proyecto. En el caso de que así sea, se procede a confeccionar el proyecto técnico como mismo se describe en la actividad: Confección del proyecto técnico del flujo alternativo anterior. Posteriormente se notifica evento a la fase de Conceptualización Arquitectónica que produce como resultado la arquitectura de dominio correspondiente. La Arquitectura de Dominio que contempla los elementos comunes de los planos temáticos y tecnológicos de abstracción a desarrollarse.

En el caso de que no sea factible desarrollar el proyecto se procede a la actividad: Archivar razones de denegación, detallada en el flujo alternativo anterior, finalizando así el proceso.

Artefactos:

- Proyecto técnico (ver Anexo 17)
- Contrato (Anexo 18)
- Acta de denegación (fecha, responsable, motivo)
- Estudio de factibilidad (Anexo 19)

2.3.2 Fase de Conceptualización Arquitectónica

Subproceso Conceptualizar Arquitectura

Provee la representación de alto nivel de abstracción del sistema, constituida por las partes del mismo, el mecanismo de integración e interrelación, restricciones y configuraciones a entornos de solución, los principios del diseño y evolución así como el ambiente tecnológico de implantación (32).

Cabe mencionar que se puede utilizar cualquier modelo de referencia para el desarrollo arquitectónico de sistemas de software, pero la presente investigación señala usar el propuesto por la Dirección Técnica de la Universidad de las Ciencias Informáticas (UCI). De cualquier forma el modelo arquitectónico siempre debe proveer un grupo de salidas, que constituyen elementos claves para seguir con la ejecución de las demás fases. Este grupo de salidas conforman el arquetipo arquitectónico o arquitectura de dominio y se describen en (Anexo 20)

El modelo de referencia para el desarrollo arquitectónico de sistemas de software propuesto por la UCI se centra en una formalización estructural de las vistas arquitectónicas que conforman la solución de software. Las mismas están constituidas por los artefactos requeridos, las restricciones tecnológicas o de diseño a valorar y una taxonomía preliminar de escenarios arquitectónicos tipos (32). Las vistas planteadas son: vistas de procesos, vista sistema, vista datos, vista integración, vista tecnología, vista presentación, vista seguridad y vista de despliegue e infraestructura. (32)El modelo arquitectónico representado en el (Anexo 21), muestra las comunicaciones entre las vistas, así como los elementos que una vista provee a las otras.

Arquitectura de Dominio: proporciona la representación de alto nivel de abstracción de los elementos comunes tanto tecnológico como del entorno de negocio al que va encaminado la LPS, que definen la línea base de la arquitectura. Esta Arquitectura de Dominio entrega a la fase de Desarrollo de Dominios la Arquitectura Corporativa del entorno de negocio objetivo. La Arquitectura Corporativa está soportada por la arquitectura de proceso, su abstracción en la vista de sistema, vista de datos y vista integración (32). Los elementos abstractos definidos en el plano de sistema constituyen el plan de producción de la fase de Construcción de Dominio.

La Arquitectura de Dominio especifica los tipos de componentes a desarrollar, clasificándolos en Componentes tecnológicos o Componentes temáticos especializados.

La Arquitectura Tecnológica inferida para la LPS define las especificaciones de los Componentes tecnológicos que soportan el proceso productivo y constituyen base tecnológica de esta. Estos se refieren a los elementos técnicos de la infraestructura como son framewoks tecnológicos o componentes partes de estos, ejemplo: reporteador, réplicador, MVC, trazas, entre otros.

Por otro lado la Arquitectura de Sistema, Datos e Integración conceptualiza las especificaciones de los Componentes temáticos especializados. Estos se refieren a los elementos de software

(componentes) asociados a áreas o procesos de negocios del segmento de mercado señalado por la LPS y conceptualizado en la Arquitectura de Dominio temática. Ejemplos de tipo de componente son: subsistema de recursos humanos, módulo de nómina, componente de monedas contables, etc.

Arquitectura de Aplicaciones: proporciona la vista de abstracción de los elementos del dominio y los elementos de variabilidad de la personalización de los productos a clientes específicos. La Arquitectura de Aplicaciones es la instanciación de la Arquitectura de Dominio tantas veces sea necesaria la personalización de los productos. Provee a la fase de Desarrollo de Aplicaciones el Plan de Producción de Aplicaciones.

El (Anexo 22), muestra la relación existente entre Arquitectura de Aplicaciones y su trazabilidad con respecto a la Arquitectura de Dominio. En la Arquitectura de Domino se definen los activos básicos que luego pueden ser instanciados en la Arquitectura de Aplicaciones, indicando las variabilidades que vienen dadas en la personalización a productos que respondan a un cliente determinado. La Arquitectura de Aplicaciones puede suprimir componentes, agregar nuevos componentes, agregar funcionalidades a componentes existentes, cambiar la forma en que se relacionan y comunican los componentes, entre otras acciones que se desprenden de la adaptación del producto al cliente.

Por la importancia que reviste como elemento rector para la fase de Construcción de Aplicaciones se detallan los conceptos de especificación que regirán la descripción de variabilidad de una Arquitectura de Dominios.

La variabilidad puede ser definida como la diferencia funcional dentro de los productos de una Línea. (56)La presente investigación clasifica la variabilidad en dos conceptos fundamentales:

Variabilidad Opcional: corresponde a las variaciones que permiten incluir varios caminos de solución a un rasgo o funcionalidad específica, las que son implementadas y por tanto provistas desde el activo núcleo y evolucionada en la medida que aparezcan nuevos casos de variación.

Ejemplo: en un componente de “Control de registros contables” una de las funcionalidades o rasgos que pueden incluir variaciones opcionales la constituye “Generar comprobante contable”. Los mecanismos de solución de variación incluyen tres alternativas:

- Generar comprobante para una sola moneda (método estándar)
- Generar comprobante con doble moneda en la contabilización (caso cubano)
- Generar comprobante a partir de patrón contable (el comprobante posee predefiniciones de cuentas a partir del proceso y documento primario que genera la transacción). (57)

Variabilidad obligatoria: define el comportamiento núcleo y propone interfaces de contrato para su extensión en los dominios de aplicación. Obligando a los ensambladores o ingenieros de aplicaciones a implementar las interfaces para dar cubrimiento a los escenarios de variabilidad,

puesto que el activo solo implementa el núcleo operacional del componente, pero queda en estado no funcional hasta tanto no le son provistos las implementaciones de las interfaces genéricas que por contrato arquitectónico fue prefijada en este para las funciones o rasgos de variabilidad.

Ejemplo: componente para procesamiento de gráficos. El componente implementa un núcleo matemático que soporta los métodos para graficar objetos y provee como elemento de generalidad, interfaces que abstraen los tipos de objetos gráficos, permitiéndose extender la variabilidad de los objetos desde los contratos que definen su forma, color, dimensión entre otros rasgos de cada tipo. Luego en la construcción de las aplicaciones se implementan los puntos de extensión (interfaces) según las necesidades del entorno del cliente. Cada aplicación esta obligada a implementar las interfaces de los objetos concretos para poder reutilizar el componente de procesamiento de gráficos en su contexto.

Los elementos de variabilidad en un plano arquitectónico para el presente trabajo lo constituyen:

- Conceptos asociados a procesos: definido cuando existe un cambio en los contratos de los conceptos de negocio, que puede estar dado hasta su nivel más atómico (actividades o tareas). Este tipo de variabilidad se puede implementar mediante herencia o uso de tipos (interfaces).
- Requisitos: definido cuando un requisito o parte de este, varía su comportamiento.
- Componente: definido cuando se requiera cambiar los contratos del mismo, generalmente causado por variación de los conceptos o requisitos.
- Reglas de negocio: definido por variaciones en los mecanismos reguladores del negocio.

La variabilidad puede estar expresada por adiciones, omisiones y cambios al plano arquitectónico de los elementos anteriormente descritos.

Especificación de la Arquitectura de Aplicaciones: constituye un elemento de referencia e instanciación de la Arquitectura de Dominios. Toma como formato el especificado en el Arquetipo Arquitectónico indicando los puntos de variabilidad. El (Anexo 23), muestra un ejemplo de cómo se define la Arquitectura de Aplicaciones (específicamente el plano de procesos, los demás planos se realizan de la misma manera) a partir de la referencia a la Arquitectura de Dominios y las variaciones del entorno de negocio específico. En la Arquitectura de Aplicaciones se identifican los elementos variables por la referencia a la Arquitectura de Dominio, con un código identificador de la variabilidad. Las adicciones de activos se formalizan siguiendo la misma taxonomía que propone el Arquetipo Arquitectónico, estos se implementan en la fase de Construcción de Aplicaciones y terminan en el repositorio como los demás activos de la línea elaborados en la Construcción de Dominios, con la peculiaridad de que ellos fueron construidos para dar respuesta concreta a un cliente específicos. Las eliminaciones no requieren ninguna especificación.

Especificación de variabilidad: describe según normas de especificación, los elementos de

arquitectura que varían señalados por la Arquitectura de Aplicaciones, siguiendo el formato establecido en el Arquetipo Arquitectónico. La variación se registra y documenta a partir de la referencia a la Arquitectura tipo y un código identificador en un documento de registro de variabilidad, que se anexa al documento de Arquitectura de Aplicaciones.

Especificación de variabilidad: [Código de variabilidad], [Ref. elemento arquitectónico que varia], [Descripción]

Ejemplo de formalización de una variabilidad: CÓDIGO: VAR-01-8/10/2011, REF: AP-02|AF-01|P-02|ACT-02, Descripción de la actividad…

2.3.3 Fase de Ingeniería de Dominios

Subproceso Construir Dominio

Se encarga de construir los elementos comunes del tipo de soluciones que pueden construirse a partir de la línea de productos definida en la arquitectura tipo o de dominio, para la familia de aplicaciones.

Los activos de software a construir en esta fase pueden ser tecnológicos o temáticos especializados, cuyo concepto fue explicados anteriormente. Esta iniciativa se conoce en el modelo de desarrollo basado en componentes por componentes reutilizables. Estos pueden ser liberados, desplegados e instanciados y ofrecen servicios a través de sus interfaces. (16)

El proceso Construcción de Dominios depende de tener realizada la fase de Conceptualización Arquitectónica, específicamente la Arquitectura de Dominio. Es la arquitectura de Dominio la que define las líneas de producto o familias de producto desde el plano abstracto de la arquitectura, constituyendo este el plan de producción de dominio de cada LPS.

El (Anexo 24), muestra la trazabilidad de la Arquitectura de Domino a la Construcción de Dominios. Los activos de software que son construidos en el Desarrollo del Dominio se entregan al Repositorio de Software o Repositorio de activos, desde donde son instanciados posteriormente por la fase de Desarrollo de Aplicaciones.

El (Anexo 25) grafica un ejemplo de trazabilidad de las vistas arquitectónicas mediante una matriz de proyección de equivalencia entre la Vista de Procesos y la Vista de Sistema para este ejemplo. Donde las áreas de procesos en la Vista de procesos están asociadas a subsistemas en la Vista de Sistema. Análogamente áreas funcionales se asocian a módulos, procesos a componentes y actividades a requisitos (32). Las áreas de procesos representan las líneas de producto o unidades de negocio de la LPS.

La metodología de desarrollo de software que se propone usar para el desarrollo de los componentes pertenecientes a cada LPS es SCRUM, explicada en el capítulo 1. (58) El modelo propuesto define que cada unidad de negocio abstrae un área de proceso, que debe ser abstraída

por la arquitectura de dominio, las partes definidas en este plano arquitectónico constituyen el Product Backlog para el desarrollo de activos básicos de la línea de producto y por consiguiente el plan de proyecto de la unidad de negocio, el que debe ser gestionado según la metodología SCRUM. La Ilustración 8, muestra la representación del desarrollo de los activos que conforman la línea base de cada LPS usando SCRUM, mediante un ejemplo hipotético. La lista de requisitos priorizada del subsistema X constituye el Product Backlog de la línea de producto. Para este Product Backlog del subsistema X se planifican 3 sprint. El primer sprint desarrolla el requisito 1, del componente 1, del módulo 1. Luego para el segundo sprint se prevé la construcción del requisito 2 del componente 1, el refinamiento de dicho componente y la construcción del requisito 2 del componente 2, terminando así la construcción del módulo 1. Por último se ejecuta el sprint tercero, que se encarga del desarrollo del módulo 2, con su componente único y sus requisitos 1 y 2.

Lo anteriormente ejemplificado arroja que la planificación de los sprints pertenecientes a cada Product Backlog se planifica y ejecuta según las especificidades y necesidades de cada unidad de negocio. La idea fundamental consiste en entregar al final de cada sprint un incremento del producto concreto y al concluir el Product Backlog entregar al Repositorio, los activos de software que se construyeron a partir de lo definido por el plano arquitectónico.

Entre los factores que intervienen en la determinación de cada Product Backlog y sus sprints están:

- Complejidad: se refiere a la dificultad técnica o de negocio a enfrentar para el desarrollo.
- Equipo de desarrollo: tiene que ver con la capacitación y el conocimiento del equipo para enfrentarse la tarea. También influye la disponibilidad de tiempo, cantidad de personas, carga de trabajo, ect.
- Riesgos: situaciones inesperadas internas o externas que influyen en el desarrollo.
- Entorno de negocio y solución: se refiere a condiciones cambiantes del entorno de negocio, la disponibilidad, confiabilidad y adquisición de tecnologías y herramientas.

Pueden existir otros factores que determinen la definición del Product Backlog, pero lo fundamental es mantener el trinomio de la gestión de proyecto: costo, tiempo y calidad, lo mas optimo posible.

El proceso inicia con la construcción de cada sprint. La lista de tareas tipos propuestas a desarrollar en cada sprint para producir los incrementos del producto se divide en cuatro flujos de trabajo fundamentales: Construcción del Activo, Integración, Prueba y Catálogo y Almacenamiento; como se muestra en (Anexo 26).

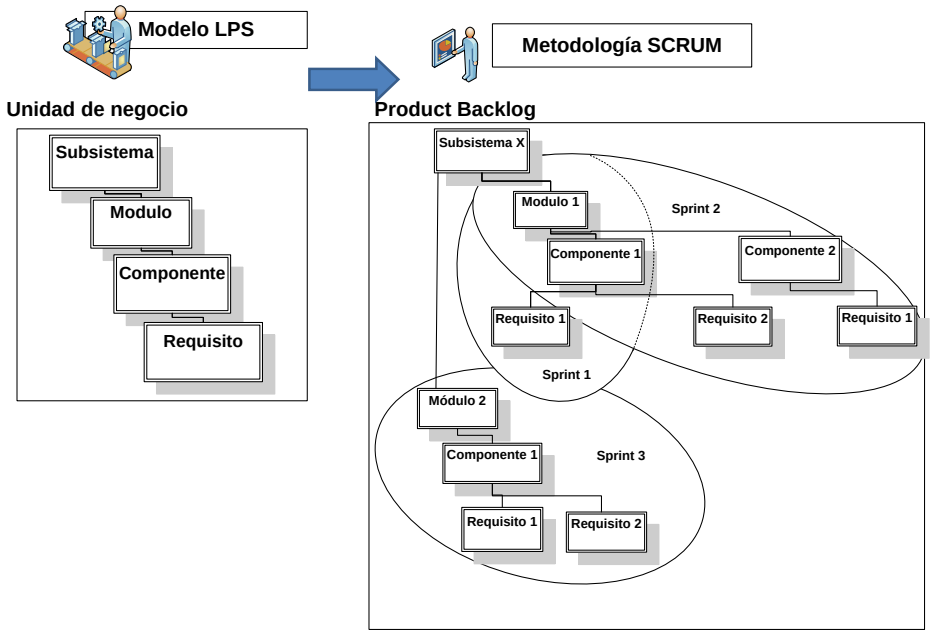


Ilustración 8 Representación del desarrollo de la LPS usando SCRUM

Construcción del Activo: Se refiere a la construcción de cada activo básico miembro del dominio.

- Especificación de escenario: aunque de la Arquitectura de Dominio se reciben los requisitos e información sobre ellos, como por ejemplo: proceso al que pertenecen, restricciones, tecnología, entre otros; es necesario detallar a un mayor nivel de profundidad estos requerimientos, para un mejor entendimiento del contexto. Para realizar una completa especificación de escenarios es necesario realizar un grupo de tareas que permitan obtener un mejor nivel de detalles, siendo estas las que se muestran en (Anexo 27). El (Anexo 28), muestra un resumen de los artefactos que se emplean en cada una de las actividades.
- Diseño: formaliza el diseño del activo a los distintos niveles de abstracción (presentación, negocio, datos, etc). Permite organizar el trabajo y aclarar aspectos conceptuales al desarrollador. En este proceso se realizan las tareas que se muestran en el (Anexo 29). El documento que más se utiliza es el de Especificación del componente, el cual concentra toda la información relacionada con las temáticas técnicas que definen al activo.
- Implementación: consiste en el desarrollo del activo usando la tecnología y herramientas señaladas. Es donde se construye el código fuente de la lógica a procesar. Entre las actividades fundamentales que se realizan en este procesos se encuentran las que aparecen representadas en el flujo de procesos que muestra el (Anexo 30).
- Prueba unitaria o automática: es la encargada de probar el comportamiento del elemento desarrollado, propiciando juegos de datos y evaluando el resultado de su ejecución con respecto a lo esperado.

Definir Integración: delimita los mecanismos de integración de cada activo específico a su nivel

más atómico; y la forma en que éste se relaciona con los demás activos de su familia de productos, cumpliendo los dictámenes realizados en la Arquitectura de Dominio.

- Ajuste y configuración de integración: el documento de Arquitectura de Dominio ya contempla los elementos fundamentales de integración de la familia de productos, en esta actividad es donde se implementan y ajustan estas especificaciones.
- Implementación de interfaces basadas en el estándar de integración definido: se implementan las interfaces de integración siguiendo el estándar definido por la Arquitectura de Dominio, que establece la conexión de los componentes de la familia de productos.
- Implementación de interfaces y estándar de interoperabilidad: se implementan las interfaces usando el estándar de interoperabilidad señalado por la Arquitectura de Dominio, que establece la forma en que los activos una vez integrados pueden comunicarse (intercambiar información).

Probar: proceso mediante el cual se conoce el estado del activo de software una vez construido, así como las condiciones que posee este para ser liberado y usado, con el objetivo de mejorarlo en caso de que el resultado no se corresponda con los requisitos planteados, lo que conlleva remitir el proceso de Construcción de Dominio al primer flujo para su rectificación. Las pruebas que se proponen realizar van encaminadas a dos objetivos fundamentales: validar que funcione el activo y verificar que lo haga correctamente.

(Realizadas con vista a validar que funcione)

- Diseño de caso de prueba: se definen y configuran los escenarios de prueba, se preparan juegos de datos, de manera que se pre-condicionan los resultados que se esperan luego de ejecutar el caso de prueba.
- Aplicación de caso de prueba: se aplica lo orientado por la actividad anterior y se obtiene un resultado con el que se conoce el estado del funcionamiento del elemento probado, pudiendo concluir si es necesario remitir el activo a la primera fase En el caso de que el activo no funcione o se detecten errores al ejecutar la prueba, se remite el proceso a la primera actividad del flujo de trabajo.

(Realizadas con vista a validar que funcione correctamente)

- Diseño de prueba funcional: a diferencia de las actividad Diseño de caso de prueba, esta busca no solo que funcione, sino que su funcionamiento se certifique funcionalmente y de la manera más optima posible. En similitud con la actividad Diseño de caso de prueba, también se señalan los juegos de datos, pero estos son evaluados por los expertos funcionales.
- Preparar juego de datos funcionales: se prepara el expediente de prueba funcional y los juegos de datos son validados.
- Aplicar prueba funcional: se aplican las pruebas diseñadas, con el juego de datos escogido y se obtiene un resultado con el que se conoce el estado del funcionamiento del elemento probado,

pudiendo concluir si es necesario remitir el activo al primer flujo de trabajo o si el activo queda certificado funcionalmente.

Definir catálogo y almacenamiento: encargado de empaquetar y delimitar el componente desarrollado dejándolo listo para almacenar en el Repositorio de activos. Pretende recoger toda la documentación que lo especifica, identifica y describe.

- Especificación del componente Cost: se especifica y documenta el componente con respecto a su mecanismo de integración, a su historia o registro de uso, clasificación, identificación, responsabilidad funcional, manuales y demás temáticas técnicas como describe el artefacto Especificación del componente.
- Empaquetado del componente: consiste en encapsular el código fuente en un paquete que pueda ser fácilmente manejado, transportado y utilizado.
- Confección de un caso de estudio: se encarga de documentar la experiencia adquirida del uso del activo, explicando cómo se emplea, que resultados se obtuvieron, cual es el punto en el que brinda mejores índices de respuestas, qué lecciones se aprendieron y cuales situaciones negativas se detectaron para su posterior evolución.
- Confección del manual: se confeccionan los manuales a todo los niveles, tanto de configuración, instalación, desarrollo o de usuario. El objetivo es proporcionar una documentación que explique y guie a los usuarios del activo.
- Establecer adición al repositorio: se entrega el componente desarrollado al Repositorio de Activos de Software cumpliendo todas las normativas y estándares definidos por este; teniendo en cuenta además su documentación y la política de versionado. El flujo de actividades señala un evento que envía la solicitud de entrada al repositorio.

Artefactos:

- Reglas de negocio →(Arquitectura. Plano proceso. Reglas de negocio) (ver Anexo 20)
- Especificación de requisitos → (Arquitectura. Plano de sistema. Especificación de requisitos funcionales de la arquitectura de sistema) (ver Anexo 20)
- Diccionario de conceptos →(Arquitectura. Plano de sistema) (ver Anexo 20)
- Mapa conceptual→ (Arquitectura. Plano proceso. Mapa conceptual) (ver Anexo 20)
- Registro de solicitudes de cambio (ver Anexo 32)
- Especificación del componente (ver Anexo 31)
- Registro de no Conformidad (ver Anexo 32)
- Casos de prueba (a considerar)
- Caso de estudio (a considerar)
- Manuales (a considerar)
- Solicitud de entrada al repositorio (a considerar)

Subproceso Repositorio de Componentes

El elemento central de la política de reutilización del modelo lo constituye el Repositorio de Activos de Software. El mismo presenta cuatro características fundamentales, la primera constituida por el estándar de formalización de los activos a reposicionar, normándose con esto la documentación técnica a utilizar para catalogar y almacenar los activos. La segunda definida por la política de versionado y los instrumentos de registro de control de los movimientos de los activos. Como tercera característica se presenta el procedimiento de entrada de los activos al repositorio y por última el procedimiento de salida de estos. El Anexo 33, muestra la secuencia de actividades de los procesos de entrada y salida al repositorio.

Flujo de Trabajo de Entrada: describe la secuencia de actividades a ejecutar para la incorporación de un componente al Repositorio de Activos. Estos componentes provienen del subproceso Construir Dominio o Construir Aplicación.

- Recibir entrada de activo: el proceso de entrada recibe la Solicitud de entrada al repositorio y la Especificación del Componente proveniente del subproceso Construir Dominio o Construir Aplicación.
- Aplicar revisión técnica formal (RTF) de estándares de arquitectura: aplicar las Listas de Chequeo de Arquitectura (estándar de codificación, taxonomía arquitectónica, principios de diseño y usabilidad por solo citar algunos ejemplos) al activo de software candidato a reposicionar. El objetivo fundamental va encaminado a garantizar las normas taxonómicas, los lineamientos arquitectónicos del componente, así como su respectiva correspondencia con la línea base tecnológica, impidiéndose de esta manera almacenar componentes con errores arquitectónicos o tecnológico, garantizándose, estandarización técnica, seguridad, cohesión, entre otros aspectos comunes regidos por la disciplina de la arquitectura. Cuando se detectan no conformidades en la RTF se rechaza la entrada del activo al repositorio, siendo enviada una orden de mantenimiento a la fuente de procedencia.

En el caso de que se corresponda lo planteado por las Listas de Chequeo de Arquitectura durante la RTF, se procede a aplicar una RTF de Especificación del Componente.

- RTF de Especificación del Componente: busca comprobar que la Especificación del Componente candidato a reposicionar está en concordancia con lo establecido en la Lista de Chequeo de Especificación del Componente. Comprobándose mediante esta, que la formalización de las características del activo es suficiente y está correctamente conformada. Es una actividad que apoya el control de la calidad de la entrada de cada activo al repositorio. Cuando se detectan no conformidades de la aplicación de la RTF se rechaza la entrada del activo y se emite una orden de mantenimiento al subproceso de donde este procede.

En el caso de que se complementen ambos pasos se pasa el flujo Ejecutar Prueba Unitaria

Funcional.

- Ejecutar Prueba Unitaria Funcional: se aplica con el objetivo de verificar que el componente se ejecuta correctamente acorde a los elementos funcionales. El resultado muestra el estado del funcionamiento del elemento probado, pudiendo concluir si es necesario rechazar el activo y emitir una orden de mantenimiento al subproceso de donde este procede, con el registro de no conformidades o solicitudes de cambio detectados.

En el caso de que los resultados de la Prueba Unitaria Funcional muestren un funcionamiento correcto se continúa con la actividad Versionar.

- Versionar: conociste en identificar numéricamente los activos de software para que puedan ser copiados, modificados y actualizados bajo un esquema de organización, que permita que estas acciones sobre los activos no se pierdan y puedan ser recuperados. Esta actividad debe desarrollarse según la política de versionado explicada posteriormente en este subproceso.
- Actualizar catálogos de componentes del repositorio: el catálogos de componentes del repositorio constituye un instrumento que permite contabilizar las operaciones de entrada y salida de un activo, modificaciones, versiones, costo, lugares en que han sido reutilizado o comercializado, constituyendo una bitácora de uso del componente y un directorio de recursos del repositorio. En el catálogo se clasifican y agrupan los activos en tres categorías:
    - Activos de infraestructura, constituida por los componentes o activos que conforman la base tecnológica o arquitectura tecnológica de la organización productiva.
    - Activos núcleos de software, constituidos por los elementos que conforman la línea base de la arquitectura de dominio, por consiguiente constituyen componentes temáticos especializados, y las principales fuentes de reutilización.
    - Activos de aplicación, constituidos por los elementos que conforman la línea base de la arquitectura de aplicaciones. Aquellos componentes modificados que constituyen ramas o variaciones de componentes de dominio, o nuevos componentes creados.
- Incorporar a Línea Base del Repositorio: realiza la incorporación del activo a la Línea Base correspondiente como parte de los componentes listos para ser ensamblados y/o reutilizados en las aplicaciones personalizadas. De esta forma concluye el flujo de actividades de entrada al repositorio.

Flujo de Trabajo de Salida: se encarga de las actividades que se ejecutan en la salida de un componente del repositorio. Los componentes son extraídos con el objetivo de desarrollar aplicaciones personalizadas para clientes específicos. Las salidas marcan las fuentes de aprovisionamiento para los procesos de ensamblaje promoviéndose con esta actividad la reutilización.

- Presentar elementos de identificación: constituye una orden o pedido de aprovisionamiento

para el flujo de Ingeniería y Construcción de Aplicaciones, en la que se describen elementos técnicos, funcionales o identificadores concretos de componentes necesarios a reutilizar.

- Buscar componente: se busca el componente en el catálogo, obteniendo también toda la documentación que lo especifica tanto individualmente como en su relación de integración e interoperabilidad con los demás miembros de la familia.
- Actualizar el catálogo de componentes del repositorio: Se actualiza el catálogo especificando los datos de la entrega, es decir, fecha, hora, uso, destino, precio, entre otros elementos relacionados con la contabilidad de la reutilización del componente.
- Entregar componente: se entrega el componente al equipo para su reutilización. De esta forma finaliza el flujo de actividades de salida.

Política de Versionado: constituye el procedimiento de control y contabilización de la evolución de las Líneas Bases definida en el repositorio de componentes o activos de la empresa. Estas Líneas Bases están asociadas a las distintas LPS definidas en la Arquitectura de Dominio de la organización productiva.

La versión de los activos se denota numéricamente por tres elementos como muestra la Ilustración 9. El primero hace alusión a la Línea Base de la arquitectura en la que el componente está situado. El segundo refiere la cantidad de variaciones que ha tenido el componente sobre la misma Línea Base (Branch).Por último el tercer elemento señala la cantidad de iteraciones devenidas de revisiones de calidad o mantenimiento sufridos por este durante el proceso de liberación o explotación experimental.

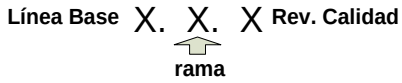


Ilustración 9 Denotación de la política de versionado

El presenta trabajo propone presentar la política de versionado enfocado en tres dimensiones. Una primera dimensión abarca lo que se denominara la Línea Base matriz, constituida por los elementos de configuración de software(componentes de los activos de software) definidos en la Arquitectura de Dominio, conformándose el control de contabilización de la base tecnológica especializada de la fábrica de software de las distintas LPS instanciada en esta.

El segundo nivel de abstracción define la Línea Base variada o ramificada de la Línea Base matriz, constituida por los elementos de configuración de software (componentes ramificados del dominio de aplicaciones) identificados en las distintas Arquitecturas de Aplicaciones identificadas para entornos de clientes concreto, conformándose en esta capa de abstracción el plano de configuración y variación de una Arquitectura de Dominio mediante su transformación en Arquitectura de Aplicaciones.

El tercer nivel de abstracción queda delimitado a nivel de activos de software. Teniéndose en

cuenta la fortaleza que implica contabilizar la evolución e índice de reutilización de estos en un entorno de producción, ya que permitiría controlar y analizar el valor neto de ingreso, el costo de producción y por consiguiente el margen de utilidad que el mismo deja, a partir de las distintas instanciaciones de este en Líneas Bases de aplicación o bien mediante la variación del mismo en otros dominios de aplicación.

En sentido general la política de versionado a nivel de Aplicación o Dominio se comportaría igual que en el nivel de activo de software, ya que el control de composición tecnológica de los activos de software que conforman determinada Línea Base, ya bien sea de Dominio o de Aplicación queda administrado por el registro de activos del Repositorio de Componentes. El Anexo 60 muestra un ejemplo detallado de aplicación del patrón de versionado anteriormente descrito.

Artefactos:

- Lista de Chequeo Arquitectura (Lo constituye las normas definidas para la especificaciones definidas en el modelo arquitectónico ver Anexo 20)
- Lista de Chequeo Especificación (Lo constituye las normas definidas para una especificación del componente ver Anexo 31)
- Catálogo (ver Anexo 56)
- Orden de Mantenimiento (a considerar)
- Caso de Prueba (a considerar)

2.3.4 Fase de Ingeniería de Aplicaciones

Subproceso Construir Aplicaciones

La Ingeniería de Aplicaciones es la encargada de implementar los productos de software que responden a clientes concretos a partir de los activos de la línea, construidos anteriormente como resultado de la Ingeniería de Dominios, que se encuentran en el Repositorio de activos.

El subproceso Construcción de Aplicaciones recibe la Especificación de variación y la Arquitectura de Aplicaciones de la fase de Conceptualización Arquitectura. El flujo de actividades se muestra en el (Anexo 34).

- Analizar Arquitectura de Aplicaciones: se analizan los documentos proveídos por la Arquitectura de de Aplicaciones. Pudiéndose identificar los elementos que son instanciados, variados y adicionados.

Caso 1: Flujo alternativo Adición:

- Enviar adicción a la fase de Construcción de Dominios: los elementos que requieren ser adicionados no se desarrollan en esta fase. La actividad sugiere ejecutar un evento de mensaje a la fase de Construcción de Dominio para que la unidad de negocio encargada de construir este activo, realice las actividades pertinentes al desarrollo del mismo. La actividad envía la Especificación del activo. El flujo de ensamblar el producto debe aguardar por la culminación del

activo para completar la construcción del nuevo producto.

Caso 2: Se requiere extraer activos para implementar su variación en una nueva rama:

- Pedido del activo al repositorio: tiene como objetivo extraer los activos del repositorio, ejecutándose en el mismo, el flujo de procesos de salida, devolviendo además del activo señalado, toda la documentación que fundamenta el mismo.

Flujo alternativo variación:

- Especificación de Variación: se recibe de la Arquitectura de Aplicaciones la especificación de variación del activo donde fueron especificados los elementos variantes. Obteniéndose elementos que especifican la variación así como otros datos del activo base del que se parte tales como proceso al que pertenece, restricciones, tecnología entre otros. Sin embargo es necesario alcanzar un mayor nivel de profundidad de los requisitos variantes, para un mejor entendimiento del contexto. Para ejecutar una completa especificación de escenarios variantes es necesario realizar un grupo de tareas que permitan obtener un mejor nivel de detalles, siendo estas las que se muestran en el (Anexo 27), como mismo se realizan en la Especificación de Escenarios de la Arquitectura de Dominios.
- Diseñar variabilidad: hace alusión al diseño de las variabilidades en las clases, base de datos e interfaces de presentación. En este proceso se realizan un grupo de tareas que se muestran en el (Anexo 29), con el objetivo de ajustar los diseños, no de crearlos, excepto la actividad Diseñar integración, que por su importancia marcada para esta fase se decide realizar de forma independiente. En el documento de Especificación del componente se concentra toda la información relacionada con el activo y las variabilidades que este presenta.
- Diseño de integración: en el diseño de integración se definen las variaciones relacionadas con servicios de entrada y salidas de los diferentes elementos arquitectónicos que conforman la aplicación, así como los formatos y estándares de interoperabilidad. Constituyen además elementos de interés del diseño de integración, las características referentes a variables de entorno globales, servicios horizontales, así como los puntos o nodos de integración que soportaran mayor rigor de integración dentro del sistema de software.
- Implementar diseños de variación: consiste en ajustar el desarrollo del activo usando la tecnología y herramientas señaladas. Es donde se ajusta el código fuente de la lógica a procesar. Entre las actividades fundamentales que se realizan en este procesos se encuentran las que aparecen representadas en el flujo de procesos que muestra la (Anexo 30), además se le agrega la actividad de implementar integración, la que ajusta todos los elementos de integración del activo al entorno de variabilidad.

Caso 3: Instanciar activo pre-elaborado sin necesidad de variación: no requiere ningún tipo de actividad pues consistiría en el uso del activo tal como fue construido. Por lo que se procedería al

ensamblaje de la solución.

Posteriormente se ejecutan los flujos de Prueba, y Catálogo y Almacenamiento de la misma manera que se realizan en la fase de Construcción de Dominios, como se muestran en el (Anexo 26). De esta forma se entrega el activo validado al repositorio y se prosigue al proceso de ensamblaje. Todas las actividades que se decidan realizar sobre los activos (adicción, instanciación o variación) deben converger en un punto común para realizar el procesos de de Ensamblado que muestra el (Anexo 35).

Tareas de ensamblado del producto

- Configurar elementos de integración: consiste en configurar las variables de entorno, los proveedores de variables y servicios globales, los ficheros de transformación, enumerados y constantes, entre otros elementos presenten en las tecnologías de arquitectura de integración.
- Ajustar e implementar interfaces: consiste en ajustar e implementar las interfaces de los elementos que varían, desarrollando el funcionamiento de la lógica de negocio especifica de la personalización a los clientes.
- Implementar integración según la especificación de la arquitectura de integración: consiste en completar la integración de los elementos configurados y personalizados, utilizándose para estos los medios tecnológicos que permiten gestionar y administrar la integración en una solución de software como son: framework de inversión de control o de programación orientada a aspectos, así como los repositorios de servicios web y sus implementaciones específicas.
- Implementar interoperabilidad según la especificación de la arquitectura de integración: en este caso se implementan extensiones definidas para el modelo de aplicación, que permite proveer o consumir información con otros sistemas legados existentes en el ámbito de despliegue del producto concreto, de modo que el soporte tecnológico contractual de estos elementos de información a inter-operar, puedan ser intercambiados entre los distintos sistemas de software existentes en el despliegue. A diferencia de las tareas de interoperabilidad del dominio, en aquel se diseñan e implementan las generalidades y estándares internacionales, en este, se implementa lo específico de cada entorno de negocio, regularmente no recogido por las normas internacionales presentes en los entornos de negocio definidos por el dominio.
- Probar integración: consiste en realizar pruebas de escenarios funcionales complejos que demanden la activación de varios módulos funcionales de la aplicación, monitoreándose de esta manera los flujos de integración del sistema, quedando validado así el correcto funcionamiento sistémico de la aplicación. En el caso de que la prueba no arroje los resultados esperados se remite el flujo a la actividad Configurar elementos de integración.
- Prueba piloto: consisten en diseñar e implantar un entorno real controlado, con las misma arquitectura de hardware y telemática del entorno final de despliegue, en el que se pueda monitorear y evaluar por parte de los principales especialistas funcionales del entorno del cliente,

el correcto funcionamiento integral del producto de software listo para su despliegue. En el caso de que la prueba no arroje los resultados esperados se remite el flujo a la actividad Configurar elementos de integración.

La fase de Construcción de Aplicaciones termina con una notificación a la fase de Empaquetado y Despliegue para entregar la aplicación personalizada al cliente, en su entrono.

Artefactos:

- Especificación de adicción de activo (a considerar)
- Caso de prueba (a considerar)
- Registro de no conformidades (ver Anexo 32Anexo 32)
- Especificación de variación (ver epígrafe 2.3.2 Fase de Conceptualización Arquitectónica, parte *Arquitectura de Aplicaciones*)
- Arquitectura de aplicaciones (ver epígrafe 2.3.2 Fase de Conceptualización Arquitectónica, parte *Arquitectura de Aplicaciones*)
- Especificación de componente (ver Anexo 56)
- Caso de estudio (a considerar)
- Manual (a considerar)
- Solicitud de entrada al repositorio (a considerar)

2.3.5 Fase de Empaquetado y Despliegue

La fase de Empaquetado y Despliegue contempla los procesos independientes de Empaquetado, Liberación y Despliegue, que pretenden llevar a cabo las actividades necesarias para declarar listo el producto, adjuntado a este los elementos descriptivos y explicativos que los complementan, así como la realización de las actividades relacionadas con la puesta en marcha del producto en el entorno cliente.

Proceso de Empaquetado: consiste en canalizar acciones de terminación y formalización del producto, de modo que el mismo contenga toda la documentación necesaria para su mejor entendimiento, explotación y funcionamiento en su destino final. Cabe mencionar que este proceso se realiza a nivel de componente en la fase de Construcción de Dominios, similarmente al flujo de trabajo Catálogo y Almacenamiento, donde el resultado de este empaquetado permite documentar el activo y asociarle a este los materiales necesarios para su reutilización, y de esta manera ser entregado al Repositorio como muestra el (Anexo 26). El Empaquetado que se realiza en esta fase es a nivel de aplicación, dando respuesta a un cliente específico. El flujo de actividades comienza a partir del evento recibido por la fase de Construcción de Aplicaciones y sigue la secuencia señalada por el (Anexo 36).

Integrar documentación técnica del producto: cada activo construido en al fase de Construcción de dominio cuenta con una documentación concentrada en el documento de Especificación del Componente. Esta actividad pretende centralizar la documentación técnica de todos los activos

utilizados en el desarrollo de la aplicación cliente, para así lograr la documentación de toda la solución recogida en el documento de Especificación técnica del producto.

- Integrar manuales de usuario: del mismo modo que la actividad anterior integra la especificación de los activos utilizados, la presente actividad debe integrar los manuales de usuarios de estos activos reutilizados y realizar un manual de usuario general del producto a partir de esta información que proporciona cada activo y tomando a su vez, todas las personalizaciones realizadas al cliente específico en su entrono de explotación.
- Diseñar identidad del producto: tiene como objetivo establecer una identidad del producto tomando los elementos representativos de la empresa cliente y adecuándolos a la solución. La identidad está dado por la marca, colores, diseño, slogan, etc.
- Confeccionar materiales de entrenamiento y capacitación: consiste en confeccionar el grupo de materiales de apoyo para logar un buen entendimiento de la solución por parte del cliente. Este grupo de materiales didácticos de formación puede estar dado por: casos de estudio, videos tutoriales, conferencias, etc.
- Diseñar e implementar soporte de distribución: consiste en formalizar la solución en una envoltura de despliegue, es decir se trata de establecer el producto en sus soporte de distribución: portal web, DVD, FTP, repositorios colaborativos, etc.
- Registrar producto: consiste en formalizar el producto en la entidad certificada para hacerlo, dándole a este un valor legal y un reconocimiento social.

Proceso de Liberación: tiene como objetivo declarar al producto listo para ser entregado al cliente, realizando un conjunto pruebas que certifiquen su nivel de completado y garanticen la aceptación del mismo, como muestra el (Anexo 37).

- Diseñar caso de prueba funcional: esta prueba no solo busca que la aplicación funciones, sino también que lo haga correctamente y de la manera mas óptima posible. En esta actividad se deben señalar los datos en que el elemento debe devolver un resultado correcto.
- Preparar juego de datos funcionales: se prepara el juego de datos mencionado en la actividad anterior y se pre-condicionan los posibles resultados.
- Preparar plan de prueba funcional: pretende hacer una planificación de la manera en la se llevarán a cabo las pruebas, señalando implicados, procedimiento, tiempo dedicado, entre otros.
- Preparar entorno de prueba piloto: se trata de organizar y asegurar el entorno en donde se va a realizar la prueba piloto, con el objetivo de garantizar los detalles técnicos del funcionamiento de la aplicación.
- Instalar liberación candidata en el entorno piloto: se instala en el entorno piloto una versión completa del producto, que clasifique como candidata a entregarse como resultado al cliente final.
- Entrenar funcionales de prueba piloto: se capacita a los funcionales implicados a partir de los materiales elaborados en el proceso de Empaquetado, para que se encuentren listos con respecto

al manejo de la aplicación en el entorno de negocio.

- Realizar prueba experimental piloto: se aplican las pruebas diseñadas, con los juegos de datos escogidos y siguiendo el plan establecido, donde se obtiene un resultado, con el que se conoce el estado del funcionamiento de la aplicación en el entorno experimental piloto.
- En el caso de que la respuesta no sea la esperada por el cliente, no esté conforme, o necesite cambiar algún funcionamiento se ejecutará la actividad de:
- Mantenimiento: se estabiliza la solución al nivel que el cliente quede satisfecho con el funcionamiento de la aplicación.
- En el caso de que la respuesta de la prueba piloto sea la esperada y que se logre en la actividad de mantenimiento estabilizar la solución, se procede al siguiente flujo de actividades:
- Firma de conformidad del cliente: actividad que pretende sellar el acuerdo de satisfacción por parte del cliente, corroborando el hecho de que la aplicación responde tal como el requiere.
- Versionar producto: esta actividad tiene lugar en el Repositorio. Persigue el objetivo de llevar un registro de las versiones que se le realizan a los productos entregados a clientes concretos, significando el versionado una adición de funcionalidades y mejoras en la aplicación. El documento que centra esta información es el Registro de versión de despliegue.
- Establecer adición al repositorio activa: se entrega la aplicación al Repositorio de Activos de Software, cumpliendo todas las normativas y estándares definidos por este; teniendo en cuenta además su documentación y la política de versionado. El flujo de actividades señala un evento que envía al Repositorio el documento de Solicitud de entrada al repositorio.

Proceso de Despliegue: persigue el objetivo de colocar al producto en su entorno de uso.

El (Anexo 38) muestra el flujo de actividades que se ejecutan en la realización del proceso de despliegue.-

- Análisis de la infraestructura telemática del cliente: la actividad consiste en realizar un levantamiento de los recursos tecnológicos de la empresa cliente donde se va a desplegar la solución, con el objetivo de tener una visión del estado de la empresa para asumir la solución de software y poder ejecutar acciones que garanticen el completo funcionamiento de la aplicación en su entorno de explotación. El análisis telemático incluye recursos de redes, telefónicos, computadoras y periféricos, servidores, licencias, etc.

En el caso de que se demanden recursos telemáticos o de hardware será necesario realizar el siguiente flujo de actividades:

- Diseñar plan de adquisición: se describe cómo serán gestionados los procesos de adquisición, desde el desarrollo de la documentación de adquisición hasta el cierre del contrato. El plan de gestión de las adquisiciones puede incluir: los tipos de contratos que serán usados, la gestión de múltiples proveedores, la identificación de vendedores seleccionados, entre otros. (55)

- Ejecutar aprovisionamiento: consiste en adquirir todos los elementos señalados en el plan de aprovisionamiento, realizando para ello todas las gestiones pertinentes a la obtención de los recursos.
- Implantar tecnologías adquiridas: consiste en implantar las tecnologías señaladas en el plan de adquisición en el entorno del cliente en que se explotará la aplicación desarrollada.
- Instalar infraestructura telemática: esta actividad se realiza cuando no existen demandas de tecnologías, pero también después de haber implantado las tecnologías adquiridas. Consiste en colocar la infraestructura telemática que soporte los requerimientos de la aplicación en el entorno de uso del cliente.
- Configurar entorno de despliegue: consiste en configurar todas las variables del entorno de despliegue del cliente, de manera que se organice y se encuentre listo para asumir el producto de software. Algunos elementos configurables pueden ser: servidores, programas de software, ect.
- Instalar aplicación: se instala la aplicación en el entorno de explotación del cliente.

Paralelamente a todo el grupo de actividades anteriores se realiza la siguiente actividad:

- Diseñar plan de entrenamiento: consiste en definir un grupo de actividades para asegurar el entendimiento del personal que usará la aplicación en el entorno de negocio. Se detallan como serán las clases de capacitación, en caso de que sea necesario impartirlas, usando para estos los materiales didácticos construidos en el proceso de Empaquetado. Se realizan trabajos prácticos, usando datos reales, para logar un correcto adiestramiento de los especialistas funcionales que emplearán el sistema, con respecto a la manipulación del mismo.

Después de haber instalado la aplicación y de tener diseñado el plan de entrenamiento se prosigue a la ejecución de la siguiente actividad:

- Impartir entrenamiento: consiste en impartir la capacitación al personal, según lo señalado en el plan de entrenamiento.
- Iniciar explotación asistida: consiste en el uso de la aplicación por los clientes, una vez capacitados estos y concretados los detalles del ambiente final de producción. Se realiza la carga inicial de los datos y se proporciona al cliente un asesoramiento en su entorno de explotación.
- Retroalimentación: consiste en recopilar las experiencias adquirida en la construcción de la aplicación, para proporcionar a la LPS elementos positivos como: casos de estudio, buenas prácticas, y elementos negativos como: identificación de cuellos de botella, procesos engorrosos. Todos con el fin de lograr un mejor funcionamiento de los productos creados por la LPS y proporcionar una guía a futuros desarrollos.
- Soporte: se ejecuta el flujo de soporte para garantizar que la aplicación se mantenga disponible, en explotación y funcionando en óptimas condiciones.

Artefactos:

- Plan de entrenamiento (a considerar)
- Plan de adquisición (a considerar)
- Casos de estudio (a considerar)
- Plan de prueba (a considerar)
- Casos de prueba (a considerar)
- Registro de no conformidades (ver Anexo 32)
- Registro de solicitudes de cambio (ver Anexo 32)
- Acta de aceptación (a considerar)
- Solicitud de entra al repositorio (a considerar)
- Registro de versión de despliegue (ver Anexo 57)
- Manual de usuario →Especificación técnica producto. Manual de usuario) (ver Anexo 58)
- Especificación técnica del producto (ver Anexo 58)

2.3.6 Fase de Administración

La fase de Administración tiene como objetivo proporcionar los recursos, coordinar y supervisar el desarrollo de activos y productos. Se divide en dos áreas fundamentales: gestión técnica y gestión organizacional, explicadas en el capítulo 1. La presente investigación se soporta en la metodología SCRUM, tomando algunos conceptos fundamentales provenientes de la guía de fundamentos del PMBOOK para realizar la Administración de la LPS. (55) (31)

Las áreas de gestión que propone el trabajo, como elementos imprescindibles a abarcar para una primera fase de implementación son:

- Método de desarrollo de la LPS basado en SCRUM.
- Gestión de riesgos.
- Gestión de costos.
- Gestión de tiempo, alcance y planificación.
- Gestión de calidad.
- Seguimiento y control.

Estas áreas de gestión son administradas y controladas por la Oficina de Gestión de Proyectos (PMO). La PMO es una unidad de organización para centralizar y coordinar la dirección de proyectos a su cargo. Pone énfasis en la planificación coordinada, la priorización y la ejecución de proyectos vinculados con los objetivos de negocio generales de la organización matriz o del cliente. Las PMO pueden operar con continuidad en aspectos que van desde proporcionar las funciones de respaldo para la dirección de proyectos bajo la forma de formación, software, políticas estandarizadas y procedimientos, hasta la dirección y responsabilidad directas en sí mismas para lograr los objetivos del proyecto. (55)

Método de desarrollo de la LPS basado en SRUM

La Ilustración 10, muestra la estructura de la LPS respecto a la organización de los activos en

familia de productos que ella abarca. El gráfico muestra como la LPS se descompone en unidades de negocio orientadas a las áreas de proceso del entorno de negocio, a la vez estas unidades de negocio dividen su funcionamiento en procesos, que expresan sus funcionalidades en activos de software.

La jerarquía de roles que se propone usar en el método LPS y su relación con la aplicación de la metodología de desarrollo SCRUM se muestra en Ilustración 11. La LPS debe contar con un jefe LPS, que tenga una visión general del crecimiento del entorno de mercado y las tecnologías de la familia de productos. Por otro lado el arquitecto o grupo de arquitectos que proyectará la abstracción de alto nivel de los activos de la LPS, sus relaciones e integraciones.

En un nivel inferior se encuentra el SCRUM Master o Jefe de la unidad de negocio, encargado de gestionar el desarrollo de los activos. Como asistente de la unidad de negocio y a su vez del SCRUM Master se propone el especialista funcional, persona que posee todo el conocimiento del entorno de negocio. Como último componente de la jerarquía se encuentra el SCRUM Team, encargado de desarrollar los activos, ya sea sean componentes de negocio. Los roles fundamentales y las responsabilidades que estos presentan se muestran en (Anexo 39)

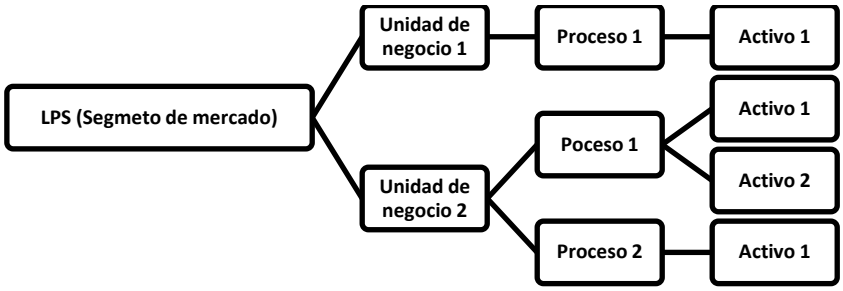


Ilustración 10 Estructura de organización de la LPS

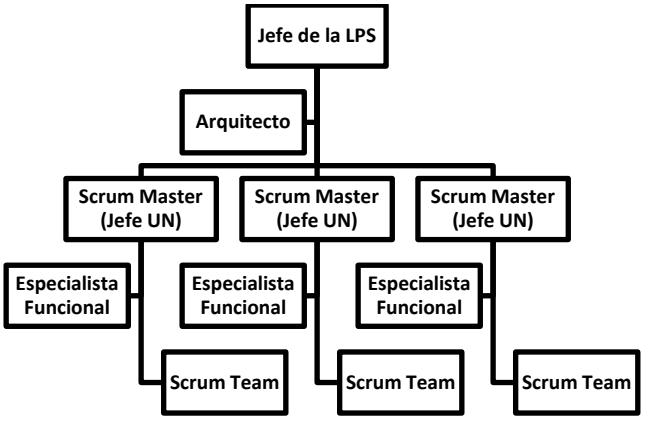


Ilustración 11 Jerarquía de roles de la LPS usando SCRUM

Como se expresó con anterioridad en la explicación del la fase de Concepción, existen dos tipos de proyectos que propone el presente trabajo: externos e internos.

La Ilustración 12, muestra la instanciación de la metodología SCRUM dentro del proceso de desarrollo de la LPS para cada tipo de proyecto. El presente trabajo define cuatro tipos de sprints fundamentales.

- Sprints de Concepción: caracterizados por las actividades definidas en la fase de concepción. El objetivo fundamental de esta es conocer si la LPS se encuentra en condiciones de enfrentar el proyecto y de ser así dar los primeros pasos para su formalización. Cuando se trata de proyectos internos se determina si es factible o no el desarrollo de nuevos activos y en el caso de que los proyectos sean externos se determina si los proceso solicitados por el cliente se complementan con el arquetipo definido.
- Sprints de construcción: este se ejecuta de una manera en la fase de Construcción de Dominio y de otra en la fase de Construcción de Aplicaciones, siguiendo las tareas tipos señaladas para cada una. En el caso de los proyectos internos se realizan en la fase de Construcción de dominios y el resultado son activos de software, tanto los que constituyen elementos tecnológicos, como los que representan componentes de negocio. Para los proyectos externos se realiza en la fase de Construcción de Aplicaciones y el resultado es un producto concreto.
- Sprints de Liberación: en el caso de proyectos internos este tiene lugar en la fase de Ingeniería de Dominios, específicamente en el flujo de trabajo Catálogo y Almacenamiento, persiguiendo el objetivo de empaquetar y delimitar el componente desarrollado listo para almacenar en el Repositorio de activos. En el caso de Proyectos externos tiene lugar en la fase de Empaquetado y Despliegue, la cual se encarga de empaquetar la aplicación que da respuesta a un cliente concreto.

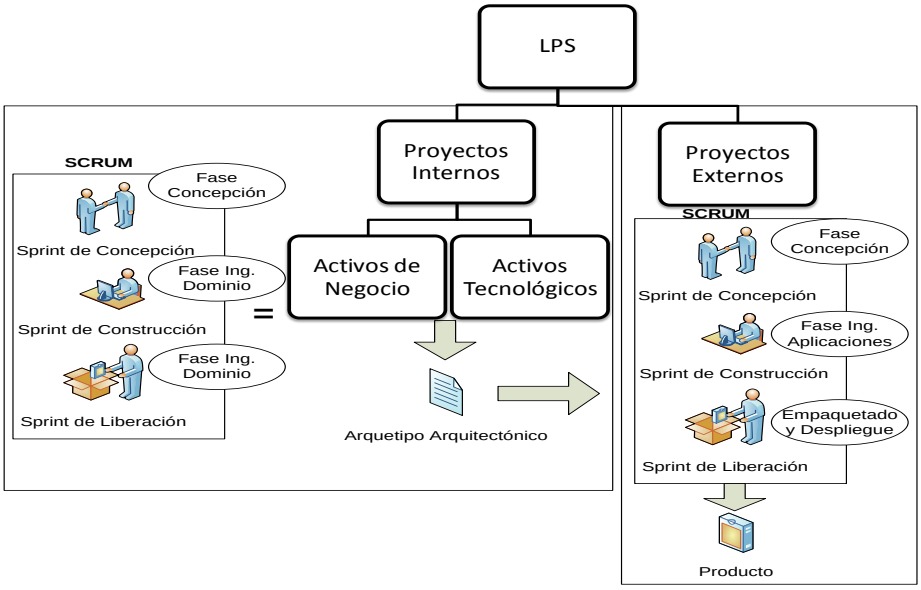


Ilustración 12 Instanciación de la metodología SCRUM dentro del proceso de desarrollo de la LPS

Un elemento de alta importancia en una metodología de desarrollo lo constituye el expediente

documental de la misma, así como las especificaciones de sus artefactos, por la importancia que reguarda en la normalización, homogenización y organización de la información. El presente trabajo propone un archivo documental unificado e integral pero que subordina su estructura organizacional a los niveles de abstracción que define el propio paradigma LPS asumido.

Los niveles de abstracción explícitos en el paradigma LPS acotados en (LPS, Línea de producto, Familia de producto) constituyen las bases de los espacios de información que se conciben en la propuesta de expediente documental. Los elementos generales de calidad, tales como listas de chequeo, definiciones y especificaciones, taxonomías de conceptos o elementos de gestión, entre otros, todos reguladores y normativos de las distintas actividades de ingeniería y gestión del proceso de desarrollo constituyen un primer espacio. Otra área de contenidos queda definida por elementos relacionados con las actividades de concepción de la LPS definidas en el flujo de trabajo de concepción. Y los otros dos espacios relacionan los elementos de ingeniería, calidad y gestión tanto del proceso de desarrollo de activos comprendido en las distintas unidades de negocio, así como los producidos en el proceso de construcción de familias de productos.

El Anexo 59 muestra una especificación detallada del expediente documental que propone el siguiente trabajo, así como las relaciones de espacio de información y los distintos artefactos. Es importante destacar que varios artefactos son repetidos continuamente en las diferentes estructuras del expediente, lo que no significa que la información quede repetida. Esto esta dado porque los procesos de gestión e ingeniería son instanciados continuamente tanto en la construcción de activos de software ejecutados en las unidades de negocio, como en el propio proceso de construcción de productos. Ejemplo de ello lo constituyen los elementos de seguimiento y control, o los propios artefactos de planificación como Product Backlog. Ambos ejemplos constituyen aspectos explícitos en los métodos de administración contemplados en la metodología, pero que deben ser instanciados tanto para proyectos que construyen activos núcleos como el caso de las unidades de negocio, como para aquellos proyectos que desarrollan familias de productos.

Gestión de Riesgos

La Gestión de los Riesgos del Proyecto incluye los procesos relacionados con la planificación, la identificación, el análisis, las respuestas y el seguimiento y control de riesgos; la mayoría de estos procesos se actualizan durante el desarrollo del proyecto. Los objetivos de la Gestión de los Riesgos van encaminados a aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos adversos para el proyecto. (55)

El (Anexo 40), muestra el modelo de procesos de gestión de riesgos propuesto. El proceso inicia aplicando una estrategia para identificar riesgos como se muestra en (Anexo 41). Este proceso determina qué riesgos pueden afectar al proyecto y se documentan sus características. Es un

proceso interactivo, ya que se descubrirán nuevos riesgos a medida que se avance con el ciclo de vida del proyecto. El flujo de actividades queda de la siguiente manera:

- Aplicar técnicas de recopilación de información, como pueden ser: tormenta de ideas, técnica Delphi, entrevista, encuesta, análisis (DAFO), análisis de asunciones, ect. (59)
- Chequear la lista taxonómica de riesgos frecuentes (ver Anexo 42): artefacto que contempla un grupo de riesgos genéricos que pueden aplicar o no al proyecto en cuestión, en dependencia de sus características.
- Aplicar el cuestionario de identificación de riesgos (ver Anexo 43) que plantea una serie de preguntas agrupadas por atributos de calidad y/o elementos en cada fase del proceso de desarrollo del software.

Una vez aplicada la estrategia para identificar riesgos se debe contar con una cantera de riesgos que serán gestionados y chequeados durante todo el desarrollo del proyecto. Luego se verifica si nuevos riesgos fueron identificados. De ser así se procede a:

- Actualizar la lista taxonómica de riesgos, dicha lista será usada en el nuevo proceso de identificación de riesgos.

En el caso de que no se hayan identificado nuevos riesgos se procede al proceso de registro.

- Registrar Riegos: se registran los riesgos en el Instrumento de Gestión de Riesgos (ver Anexo 44), de manera que se documenta toda la información asociada a este.
- Realizar análisis de los riesgos: implica las actividades de:
  - Análisis con desglose por objetivos: que equivale a la evaluación cuantitativa de los riegos, de manera que se evalúa el impacto de los mismos respecto al tiempo, el alcance, el costo y la calidad.
  - Priorización de riesgos: que busca brindar mayor atención a los riesgos con mayor probabilidad de ocurrencia.
  - Identificar respuesta a los riesgos: busca desarrollar opciones y determinar acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto. Incluye introducir recursos y actividades en el presupuesto y cronograma
  - Seguimiento y control: esta actividad se realiza en cada punto de chequeo del proyecto, garantizando el seguimiento a los riesgos para conocer su estado. Se emite un acta periódica en cada chequeo. Posteriormente se puede regresar a la actividad Realizar análisis de los riesgos para repetir el proceso en cada chequeo, durante la ejecución del proyecto.

Artefactos:

- Lista taxonómica de riesgos (ver Anexo 42)
- Cuestionario de identificación de riesgos(ver Anexo 43)

- Instrumento de gestión de riesgos (ver Anexo 44)
- Acta paródica semanal(a considerar)

Gestión de Costo

La Gestión de los Costes del Proyecto incluye los procesos involucrados en la planificación, estimación, preparación del presupuesto y control de costes de forma que el proyecto se pueda completar dentro del presupuesto aprobado. La Gestión de los Costes del Proyecto se ocupa principalmente del coste de los recursos necesarios para completar las actividades del cronograma. (55) (El Anexo 45), muestra el flujo de actividades que define este proceso.

- Identificar elementos de gasto: consiste en identificar todos los elementos que representan gastos en el proyecto tomando como referencia la lista taxonómica que contiene un grupo predeterminado de estos. Ejemplo de ellos materiales (hojas, plumones, lapiceros, ect), mano de obra (salarios), costos indirectos de producción (depreciación, teléfono, combustible, servicios, transporte, etc), así como los que representan inversiones (tecnología, software).

En el caso que se encuentre otro elemento de gasto que no esté descrito en la lista taxonómica de costo:

- Formalizar e incluir a lista taxonómica: se especifican todos los atributos de los elementos de gasto, entre estos figuran: tipo (gastos directos, gastos indirectos e inversiones), unidad de medida, fabricante, proveedor, código, etc y se adicionan a la lista taxonómica de costo.

Si no se identifican nuevos elementos de gasto en el proyecto se procede a:

- Especificar la tarifa y precio de los elementos de gasto: se señalan las tarifas y precio de cada elemento de gasto, tanto para venta mayorista como minorista. Si existen variaciones con respectos a los demás atributos del elemento de gasto, se regresa a la actividad Formalizar e incluir a lista taxonómica para actualizar los atributos que han variado.
- Crear ficha de costo: es un subproceso que se encarga de gestionar todos los costos que pueden influir en el proyecto. La ficha de costo incluye tres categorías principales y cita los ejemplos que muestra el (Anexo 46).

El (Anexo 47), muestra el flujo de actividades que ocurren al ejecutarse el subproceso Crear Ficha de Costo.

- Calcular costos directos: se calculan en el instrumento ficha de costo, los costos que influyen directamente en el proyecto, tanto de mano de obra, como por concepto de materiales e insumos.
- Calcular costos indirectos: se calculan en el instrumento ficha de costo todos los costos que no influyen directamente en el proyecto, pero si deben ser tomados en cuenta. Para los costos indirectos el proyecto representa un elemento más de la empresa según el criterio de reparto

usado.

- Calcular inversiones: es necesario calcular el costo de inversión de aquellos elementos que pueden ser necesarios adquirir.
- Análisis de factibilidad económica: consiste en calcular si el total de inversión es mayor que el 15% del total de costos directos más el total de los costos indirectos planificados.

En caso de que la inversión represente más del 15% de los costos directos más los indirectos, se procede a realizar un:

- Análisis de curva de retorno de inversión: se realiza buscando conocer el punto en que los ingresos son iguales a lo invertido inicialmente, más los gastos tanto directos como indirectos que se van generando. Es el momento en el cual se comienzan a obtener utilidades. Para el presente trabajo se considera una variante factible si se obtiene el punto de retorno de inversión en 1 año y medio como máximo. (60)

De ser factible el resultado del análisis arrojado por la curva de retorno de inversión y en el caso de que el 15% de la inversión no sea mayor que los costos directos más los indirectos, se realiza la actividad:

- Diseño de precio del producto: el análisis de precio del producto depende un conjunto de elementos entre los que figuran algunos como mercado (competencia), cantidad de ventas, costos de producción, margen de utilidad definida por el Ministerio de Finanzas y Precios (MFP), entre otros. Para el diseño del precio del producto es necesario tener en cuenta los costos netos de inversión en la realización del producto, más el margen de utilidad definido por el Ministerio de Finanzas y Precios.

En el caso de que el resultado del análisis arrojado por la curva de retorno de inversión no sea factible:

- Alertar la PMO: se emite una alerta a la dirección de proyectos para que esté al tanto y valore las posibles alternativas a tener en cuenta.

De esta manera finaliza el subproceso Crear ficha de costo y continúa el proceso de Gestión de costos con su flujo de actividades, ejecutándose seguidamente la actividad:

- Diseñar cronograma de ejecución de financiamiento del presupuesto: consiste en definir los momentos en que se realizarán desembolsos de presupuesto según el cronograma de ejecución del proyecto, que delimita el alcance y el tiempo del mismo. Está estrechamente ligado al plan de aprovisionamiento, y pretende marcar los hitos de financiamiento en el desarrollo del proyecto.
- Ejecución del costo: consiste en gestionar y ejecutar las acciones de aprovisionamiento, subcontratación o gastos asociados a los elementos de desembolso definidos en el cronograma de ejecución financiero del proyecto.

Artefactos:

- Taxonomía de gastos (a considerar).
- Instrumento de gestión de costo o ficha de costo (a considerar).
- Cronograma de ejecución financiero y presupuesto (a considerar).

Gestión de Calidad

La calidad según el presente trabajo debe tener dos enfoques fundamentales: Proceso y Producto.

Proceso: se refiere a la gestión y aseguramiento que permita hacer cumplimentar las regulaciones, estándares y lineamientos normadas por la PMO y descritos en los flujos de actividades de administración. La dimensión proceso se centra en tres elementos fundamentales:

Revisiones técnicas formales (RTF) a las políticas de planificación.

- Relacionado con la planificación

La planificación se realizará sobre la base de una métrica, la que deberá ser continuamente ajustada según mediciones que gestionan los miembros de la PMO. El Product Backlog en la fase de Ingeniería de Aplicaciones representa la entrega del producto. En el caso de la Ingeniería de Dominios, el Product Backlog se asocia a las unidades de negocio, dependiendo de una serie de variables (riegos, complejidad, recursos) explicadas en el Proceso Construir Dominio, que pueden definir nuevos Product Backlogs asociados a procesos dentro de la unidad de negocio o a sprints sencillos sin necesidad de desmembrar el Product Backlog de la unidad de negocio, en varios Product Backlogs por procesos que ella abarca.

La planificación de cada Product Backlog contará con sprints que iteraran el desarrollo en periodos de 15 a 30 días, el máximo período que puede contener un sprint es de 30 días. Cada sprint deberá ser negociado con el cliente y aprobado por este. Deben existir tres tipos de sprint: Tipo 1 Sprints Concepción, Tipo 2 Sprints de Construcción, Tipo 3 Sprints de Liberación.

- Relacionado con las tareas

De una tarea no debe dejarse de definir los elementos: descripción, tipo, fecha inicio y fin, relación con otras tareas, estado, asignación, iteración, evaluación

La planificación de las tareas debe ser personal por cada miembro. Las tareas asociadas al sistema de trabajo y que constituyen tareas periódicas dentro del ciclo de vida del proyecto, no necesitan ser reguladas dentro de los cronogramas del proyecto.

Deben existir como mínimo cuatro tipos de peticiones de trabajo:

- Tareas: asociado a las tareas que constituyen unidades de trabajo que tributan al incremento del producto de software asociado al proyecto.
- Acuerdo: relacionado con las tareas, indicaciones y acuerdos que se toman en las reuniones de

chequeo de los sistemas de trabajo, o asuntos administrativos dentro el proyecto o la organización. La lista de peticiones de tipo acuerdo constituyen el registro de acuerdos del proyecto.

- Hito: tareas especiales que no deben tener planificado tiempo, pero que deben ser asignadas en la planificación con el objetivo de marcar etapas de chequeo de resultados entregables y tangibles de un conjunto de tareas. Los hitos deben estar definidos a nivel de sprint y asociados a cada miembro del equipo.
- Errores: tareas relacionadas con revisiones técnicas formales que efectúan los miembros de la PMO de la organización de desarrollo, los clientes, o los mismos miembros del proyecto. Relacionado con Bugs y errores detectados en el producto de software o artefacto documental de este. El reporte de las peticiones de tipo error constituyen el control de Bug y traza de soluciones de error del proyecto.

Una tarea no debe exceder un tiempo de duración mayor a 3 días. En el caso que el volumen de trabajo o la complejidad de una tarea requiera un tiempo mayor, esta debe ser fraccionada en objetivos mas pequeños que permitan iterarla en sub-tareas, de modo que el conjunto de resultados de estas, constituyan la solución final de la tarea inicial.

Un cronograma debe tener hitos de control cada 3 o 5 días como máximo. Los hitos deben estar asociados a resultados generales del proyecto, pero distribuidos de acuerdo con el número de miembros del proyecto. Por ejemplo si un proyecto posee 2 miembros, este debe tener en los primeros 5 días como mínimo 2 hitos de control, uno por cada miembro.

Los hitos deben estar asociados a tipos de resultados de proyecto tales como:

- Informe de resultado: cuando las tareas se asocian a revisiones, investigación, I+D, etc.
- Taller con cliente o equipo de trabajo: cuando el resultado está asociado a trabajo de requisitos, procesos, investigación de negocio, ingeniería de requisito.
- URL o Unidad funcional de prueba de concepto: asociado a una prueba de concepto, en la que un componente o tarea de codificación ha sido terminada, lo que libera la terminación de una tarea de codificación es un resultado positivo en su prueba de concepto.
- Release: instalador de salida que constituye revisión candidata o versión final de un proyecto.
- Elemento que constituye una salida pactada en cronograma con el cliente: Cualquier otro elemento pactado con el cliente.

Revisiones técnicas formales a los sistemas de trabajo de la PMO. El (Anexo 48), muestra los principios que deben tenerse en cuenta en la definición de las listas de chequeo para el control y revisión de dichos sistemas de trabajo.

Revisiones técnicas formales a los sistemas de trabajo del equipo de proyecto de la LPS. Las características de estas deberán estar en correspondencia con la metodología de desarrollo que se

utilice. En el caso del presente trabajo (SCRUM), y funciona según se describe en el (Anexo 49).

Revisiones técnicas formales de seguimiento y control: encaminadas a regular los mecanismos de seguimiento y control en la LPS. Algunos de estos mecanismos deben tener en cuenta los siguientes elementos:

- Cada proyecto debe llevar una bitácora de acuerdos y un registro de las actas de reuniones.
- Cada equipo debe definir hitos tanto de control como de entrega.
  Entrega: son los pactados con el cliente y están en concordancia con las tareas tradicionales de desarrollo de software (instaladores, manuales de usuario, documento de requisitos, etc)
  Control: el presente trabajo define tres tipos fundamentales de hitos de control:
    - Informes: entregable asociado a tareas de investigación más desarrollo, entre los que figuran análisis de factibilidad, estudio de mercado, etc.
    - Presentación o Documentos (workshop): que se originan de talleres con el equipo de trabajado, con especialistas funcionales o clientes.
    - Casos de prueba: asociados a las tareas de implementación, lo que define que una tarea de implementación culmina si y solo si, el caso de prueba asociado a este es ejecutado y comprobado satisfactoriamente. Con este lineamiento se logra instrumentar la aplicación del principio de Integración Continua, (61) garantizándose que la industria de partes sea controlada de manera estricta y sistemáticamente.
- Los equipos de desarrollo de la LPS deben implementar acciones de auto medición que permitan ajustar las métricas a sus características y entorno. Esta acción debe ejecutarse en un rango de tiempo entre un mes y tres meses.
- Las reuniones de control se deben realizar basadas en los siguientes puntos:
    - Estado de ejecución del cronograma físico.
    - Estado de ejecución del cronograma financiero.
    - Estado de los riesgos.
    - Control del errores y solución de no conformidades.
    - Evaluación de resultados de las revisiones técnicas formales.
    - En cada reunión de chequeo que se detecte variaciones del cronograma para más o menos tiempo debe llenarse un registro de variaciones.

Producto: se refiere a las prácticas y lineamientos que buscan conseguir calidad en cada entrega o terminación de productos o partes. Sus principales exponentes son:

- Pruebas unitarias a componentes: llevadas a cabo principalmente por desarrolladores y arquitectos. Validan que los componentes estén desarrollados respetando los estándares arquitectónicos y de integración definidos. Persigue además que los componentes funcionen y cumplan con la taxonomía definida en la fase de Arquitectura. Garantiza el principio de integración continua.
- Pruebas funcionales: llevadas a cabo principalmente por los clientes y especialistas funcionales. Validan las funcionalidades a partir de reglas del negocio y escenarios funcionales,

recreados en juegos de datos pre-elaborados y evaluándose las respuestas respecto a los datos esperados.

- Pruebas piloto: prueba que se realiza en un entorno controlado de producción con la misma arquitectura de infraestructura del cliente y escenarios funcionales preconcebidos.

Las pruebas anteriormente explicadas pueden arrojar dos tipos de errores:

- Errores de codificación: asociadas a apruebas unitarias.
- Errores de no conformidad: asociadas a pruebas funcionales y pruebas pilotos.

En cada Sprint Review el cliente deberá emitir un acta de conformidad certificando que los elementos probados del sprint terminado fueron funcionalmente validados; en caso contrario, se emitirán no conformidades o solicitudes de cambio.

En el caso de las Revisiones Técnicas formales o Pruebas Unitarias, se emitirán tareas de mantenimiento tipificada con la categoría error.

Artefactos

- Registro de variaciones (a considerar)
- Registro de acuerdos (a considerar)
- Acta de aceptación (a considerar)
- Registro de no conformidad (ver Anexo 32)
- Registro de Solicitud de Cambio (ver Anexo 32)

Gestión de Tiempo, Alcance y Planificación.

Los procesos de gestión de alcance y tiempo identifican, definen y maduran el alcance del proyecto, el coste del proyecto y delimita los elementos a planificar. A medida que se obtenga nueva información sobre el proyecto, se identificarán o resolverán nuevas dependencias, requisitos, riesgos, oportunidades, asunciones y restricciones. Los cambios significativos durante el ciclo de vida del proyecto provocan la necesidad de reiterar uno o más de los procesos de planificación y estimación. (55)

La Gestión del Alcance del Proyecto presenta los procesos necesarios para asegurarse que el proyecto incluya todo el trabajo requerido, y sólo el trabajo requerido, para completar el proyecto satisfactoriamente. Se relaciona principalmente con la definición y el control de lo que está y no está incluido en el proyecto. Se compone de los procesos de dirección de proyectos: Planificación del Alcance, Definición del Alcance, Crear EDT[4], Verificación del Alcance y Control del Alcance. (55)

La Gestión del Tiempo del Proyecto incluye los procesos necesarios para lograr la conclusión del proyecto a tiempo. Se compone de los procesos de dirección de proyectos Definición de las

[4] Estructura de desglose del trabajo

Actividades, Establecimiento de la Secuencia de las Actividades, Estimación de Recursos de las Actividades, Estimación de la Duración de las Actividades, Desarrollo del Cronograma y Control del Cronograma. (55)

Para realizar la Gestión del Alcance y Tiempo de la línea se propone usar un instrumento de métrica. La métrica está orientada al proceso de desarrollo de software y no al producto. Está soportada en la metodología de trabajo SCRUM y las categorías de tareas tipo definidas en el método de desarrollo que asume la LPS. Vincula dos dimensiones fundamentales:

- Recurso Humanos: se mide el desempeño de los desarrolladores y se calcula una norma promedio respecto a tres niveles de capacidad productiva, categorizados en: personas de alta productividad, personas de media productividad y personas de baja productividad.
- Tareas tipos: identifica las tareas tipos asociadas al método de desarrollo, y parte del plano arquitectónico que provee la fase de Concepción Arquitectónica. Las tareas tipos reciben una categoría respecto al nivel de complejidad de las mismas en: tareas de alta, media y baja complejidad. La complejidad de las tareas tipos está dada por el volumen de procesamiento y la complejidad algorítmica. La tecnología no se tuvo en cuenta como indicador en los análisis de la métrica debido a que la LPS parte de una infraestructura tecnológica única y preconcebida ya desarrollada.

La definición del alcance tiene lugar en la construcción del Product Backlog cuando el cliente (Product onwer), el especialista funcional, el jefe de proyecto (SCRUM master) y el Jefe de línea de producto, definen los objetivos y requisitos del producto, así como los sprints en los que se realizarán entregas del producto. Cada sprint constituye una EDT que subdivide el trabajo del proyecto en porciones de trabajo más pequeñas y fáciles de manejar, donde cada nivel descendente de la EDT representa una definición cada vez más detallada del trabajo del proyecto. La EDT en SCRUM obtiene un mejor nivel de detalle en la reunión de planificación del sprint, donde el equipo selecciona los requisitos prioritarios señalados por el cliente y realiza una descomposición jerárquica del trabajo, para lograr los objetivos del proyecto y crear los productos entregables requeridos. Este alcance puede ser modificado en cada sprint según las variaciones del cambio de requisitos realizando por los clientes.

Cuando se trata de construcción de aplicaciones orientadas a clientes concretos la planificación parte de una EDT proveniente de la Arquitectura de Dominio y sus variaciones durante el proceso de Ingeniería de Aplicaciones, constituyendo este el Product Backlog.

Esta métrica constituye en su definición, un cronograma tipo que sirve de instrumento metodológico para la implementación de la planificación.

Seguimiento y Control

El flujo de seguimiento y control constituye el elemento estratégico que soporta la dirección

administrativa de la alta gerencia de la LPS. El mismo debe ser administrado y gestionado por la PMO, quien constituye el elemento rector y controlador del proceso de desarrollo y a su vez el sistema nervioso digital que recolecta, procesa y estructura la información para la ayuda a la toma de decisiones y el monitoreo del proceso de productivo de la LPS.

Un sistema de seguimiento y control debe partir por definir los puntos de control del proceso. Espacios que permiten establecer los mecanismos reguladores para monitorizar, contabilizar y caracterizar en planos cuantitativo el proceso productivo y su evolución. En este sentido el presente modelo define dos niveles de control para el proceso productivo. El primero definido en el sistema de trabajo de la PMO, orientado a gestionar los indicadores macro del proceso productivo, el segundo concentrado en el sistema de trabajo de los equipos de proyectos que debe contabilizar los indicadores más específicos de los productos.

El sistema de informes que debe rendir la PMO como resultado de aplicar los mecanismos de seguimiento y control y gestionar los indicadores de los diferentes espacios de control definidos en los sistemas de trabajo tanto de la PMO como de los equipos de proyectos deben estar constituidos por los siguientes elementos:

Informe de Monitoreo: orientado a representar una foto en tiempo real del proceso productivo, concentrando principalmente los indicadores de eficiencia, utilización de fuerza de trabajo, productividad y los principales riesgos presentes. La frecuencia del informe debe ser semanal, destinado a la alta gerencia y los jefes de equipos de proyectos. El (Anexo 50) muestra los principales indicadores a representar.

Informe de Resultados: orientado a representar un resumen estadístico del estado de cumplimiento de los planes productivos, los principales indicadores de eficiencia económica, así como niveles de productividad alcanzados. La frecuencia del informe debe ser mensual, destinado a la alta gerencia, los jefes de equipos de proyectos desarrolladores. El (Anexo 51) muestra los principales indicadores a representar.

Conclusiones

- El modelo centra su aporte fundamental en la introducción de un modelo de desarrollo arquitectónico, que defina el mapa de domino y por consiguiente, el plan de producción de las diferentes áreas productivas contempladas en el modelo inicial LPS propuesto por el SEI. Otro aporte significativo del trabajo radica en la vinculación de técnicas de desarrollo ágiles en las áreas de administración e ingeniería de un modelo industrial de producción de software.
- El modelo propuesto manejas los conceptos de: Ingeniería de Dominios, Ingeniería de aplicaciones, Administración y Arquitectura como disciplina que se le adhiere al modelo, con el objetivo de lograr una abstracción arquitectónica de la solución antes de los desarrollos.
- Las fases fundamentales que el modelo plantea son:

- Concepción: encargada de conceptualizar el proyecto, analizar la factibilidad y capacidades de su realización y encaminar pasos a su formalización.
- Conceptualización Arquitectónica: propone la realización de una Arquitectura de Dominio para las abstracciones de los elementos comunes de la familia de productos, y la Arquitectura de Aplicaciones para la definiciones arquitectónicas de las variabilidades, en las personalizaciones a clientes, de la arquitectura tipo definida en la Arquitectura de Dominios. Propone para la definición de la arquitectura, la instanciación de un modelo de desarrollo arquitectónico.
- Ingeniería de Dominios: encargada de la construcción de los elementos comunes de la familia y su almacenamiento y evolución en el repositorio de activos de software.
- Ingeniería de Aplicaciones: encargada de desarrollar los elementos que varían definidos a partir de los activos de software existentes y la Arquitectura de Aplicaciones, dadas estas variaciones por las personalizaciones a clientes concretos.
- Empaquetado y Despliegue: se encarga de liberar el producto, empaquetar el mismo con todos los elementos necesarios que lo complementan, y colocarlo en el entorno de negocio del cliente.

- El modelo propone un área de Administración de la LPS, que sugiere el método de desarrollo basado en SCRUM, además propone el proceso de gestión de Riesgos, Calidad, Costos, Seguimiento y Control, Alcance, Tiempo y Planificación. Se propone que en próximos desarrollos se complete la fase de Administración con los elementos de Gestión Organizacional planteados por el SEI, así como incluir al modelo actividades que contribuyan en incluir o reforzar las tareas orientadas a las personas y al desarrollo de los equipos de trabajo.
- El modelo propone la guía de actividades que describen los procesos ingenieriles y de gestión, los roles involucrados en estos procesos y los artefactos y documentos que se centralizan en un expediente digital organizado. A su vez existe la necesidad de proponer al modelo herramientas que den soporte a todo el proceso.

CAPÍTULO 3 VALORACIÓN DE RESULTADOS

Introducción

El presente capítulo realiza un análisis antes y después de la aplicación del modelo a las empresas SICS y DESOFT. Para esto se estudia la medida en que pudieron ser cubiertos los elementos fundamentales que deben contemplar las LPS, sugeridos en la explicación de los conceptos fundamentales del modelo en el capítulo dos. También se valoran los principios propuestos por Bohem y Turner (criticidad, dinamismo, personal, tamaño y cultura), donde se realiza un análisis de este enfoque para determinar el tipo de metodología (ágil o robusta) que más se adecúa a las empresas objetivos. Se realiza además un análisis del comportamiento de las variables objetivos a medir por la presente investigación, señaladas en la introducción.

3.1 Caso de estudio empresa SICS

3.1.1 Cubrimiento de los elementos fundamentales de LPS

Antes del modelo:

Dominio: encaminado al objeto social de la empresa, soluciones de software para la informatización del sector del transporte. No existían estudios encaminados a identificar y caracterizar los procesos del sector del transporte. No existía un plan de informatización ordenado para el desarrollo de soluciones de software del ministerio y las empresas del sector.

Familia de producto:

- Producto viajero: encargado de la informatización de las agencias de viajes del territorio nacional, enmarcado en la tipología de productos GDS (Geography Distribution System). El mismo presentaba un atraso en cronograma de 17 meses y un sobregiro de presupuesto de 255 000 CUP.
- Combustible: sistema dedicado a la gestión, control y monitorización de la explotación de los medios de transporte y el combustible de una empresa.
- Inventarios ociosos: sistema dedicado a la gestión de los artículos ociosos de las distintas empresas del ministerio.
- Producto Condor: encargado de la gestión contable y financiera de las empresas, enmarcado en la tipología de productos de gestión empresarial. El mismo presenta desactualización tecnológica, limitaciones funcionales con altos índices de incidencias e insatisfacciones de clientes.

Modo de producción: existía un modelo de producción basado en RUP, con procedimientos de calidad extensos y no funcionales, los desarrollos se realizaban organizados por proyectos individuales, no existían mecanismos de control ni evaluación del proceso productivo.

Arquitectura: no se conocía la disciplina de la Arquitectura de Software, no se contaba con marcos de trabajo preconcebidos y adaptados al entorno. Básicamente se utilizaban variadas plataformas, fundamentalmente Visual Fox Pro, .Net y Delphi.

Activos de software: no se hacía uso del concepto, los desarrollos no estaban orientados a la reutilización. No se identificó ningún ejemplo de prácticas o elementos de reutilización.

Después del Modelo:

Dominio: se mantuvo como dominio las soluciones de gestión de la actividad del transporte formalizándose mejor las fronteras y objetivos de informatización en un plazo de tres años**.** La aplicación del modelo arrojó como resultado la creación dos LPS. Una orientada al dominio de soluciones de gestión, planificación, distribución y comercialización de capacidades de viaje y la segunda enfocada en el dominio de soluciones de gestión administrativa de la actividad del transporte y sus insumos.

Familia de producto: se reorganiza el proceso productivo lográndose resultados históricos en el proceso de producción tanto en planos económicos y de facturación, como en los volúmenes de productos aplicados con fuerte impacto en la informatización del sector del transporte en el país.

LPS1 GDS (Soporta el desarrollo en Tecnología .Net)

- P1-Producto Viajero: se logra rediseñar la solución existente, ordenar los esquemas de reutilización y evolucionar el plano tecnológico del mismo en un ciclo productivo de cinco meses y medios, encontrándose este producto actualmente en proceso de generalización.
- P2-Sistema para la gestión de la empresa comercializadora de pasajes VIAZUL.
- P3- Sistema para la gestión de pasajes de la firma CUBANACAN

Ambos resultados se logran en paralelo, con un 89 % de los activos de software desarrollados para Viajero2 y el tiempo estimado para pilotaje pactado en 75 días, por un costo de solución de 135 000 CUP cada uno.

LPS2 Paquete de soluciones de gestión administrativa (Soporta el desarrollo en tecnología PHP)

- P4-Sistema para el control y gestión de los medios de transporte y el combustible 2.0.
- P5-Sistema para la gestión y ejecución de un presupuesto de inversiones del transporte.
- P6-Sistema para la informatización de la empresa Selecmar.
- P6-Portal corporativo para la gestión administrativa del sector empresarial del Mitrans.
- P8-Sistema para la gestión y control de puestos de mando del Mitrans.

Modo de producción: se logró implantar un 70% de los elementos contemplados en el modelo, en ambas LPS. En este caso de estudio, por la limitación de personal y la dimensión de la empresa, el 30 % de los elementos excluidos del modelo durante la aplicación del mismo lo constituyeron la, Seguimiento y Control, Gestión de Riesgo, Costo y Calidad, todos del área de Administración.

Arquitectura: se homogenizan las soluciones de desarrollo concentrando todo el proceso productivo sobre tecnología .NET y PHP, soportándose en una infraestructura tecnológica y de framework. Se deja Visual Fox Pro como tecnología para el soporte del producto Condor.

Activos de software: se desarrollan e implantan dos marcos de trabajo tecnológico que pasan a constituir la base tecnológica del proceso productivo. Uno desarrollado con tecnología. Net (propietaria) y basado en Smart Client Application, framework provisto por Microsoft.Co. Otro desarrollado con tecnología PHP basado en Joomla, CMS (Content Manager System) basado en software libre. Otros activos temáticos producidos de mayor impacto en los índices de reutilización lo constituyeron los mostrados en el (Anexo 52) y el (Anexo 53) por cada LPS.

3.1.2 Análisis de criterios según Boehm y Turner

Tamaño: el número de personas en el entorno de desarrollo de software de SICS es escaso. Los equipos de trabajo oscilan entre dos y cinco personas. Los nivel de comunicación entre este grupo de personas son sencillos. El conocimiento explícito se gestiona por medio del contacto personal, talleres, correo, entre otros, sin embargo no siempre se logra socializar el conocimiento con altos porciento de efectividad. Los proyectos son mayormente de duración entre pequeña y mediana. Por tanto se sitúa al criterio en una categoría baja.

Criticidad: no están involucradas pérdidas de vida, aunque si existe el riesgo de sucederse pérdidas de recursos, pues siempre puede retirarse algún miembro del proyecto de la entidad, perdiéndose con ellos gran parte de la especialización. También se pierden recursos monetarios para la empresa y para los trabajadores cuando no se cumple el plan, lo que da al traste con desmotivaciones personales. Con respecto al confort, vale decir que se cuenta con internet ilimitado, no se tiene repositorios de herramientas, por lo que se afrontan dificultades en obtener las tecnologías necesarias y actualizadas. Las PC y los locales no alcanzan o quedan en condiciones de hacinamiento, haciendo compleja la concentración en las tareas. Por lo anteriormente expresado situamos este criterio en una escala media-alta.

Dinamismo: los requerimientos en el entorno de desarrollo de software no son muy cambiantes, aunque si se han modificado moderadamente (cambios impuestos por la nueva política económica y social del Partido Comunista de Cuba (PCC). Sin embargo en los contextos técnicos si suceden cambios (introducción de nuevas tecnologías); incluyendo también los requisito de negocio que requieren cambios en la tecnología. Por tal motivo ubica en una escala relativamente baja-media.

Personal: El personal con que se cuenta en SICS es en su mayoría inexperto, pues sus recursos humanos son en gran parte técnicos medios e ingenieros recién graduados. Teniendo en cuenta lo anteriormente planteado se ubica el indicador en una escala alta.

Cultura: la organización de la documentación es flexible y ligera, así como el formalismo de las reuniones y las comunicaciones. Existen sistemas de trabajos y chequeos, aunque no siempre

siguen estándares, procesos y políticas de la forma correcta. Las personas tienen un margen de libertad a la hora de alcanzar los objetivos. Por tal razón ubicamos a la cultura en una escala baja.

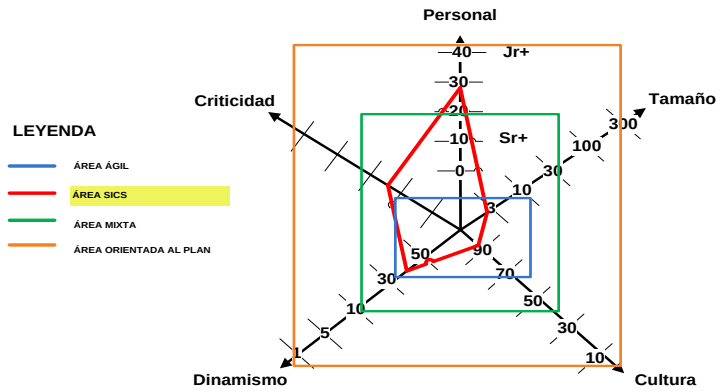


Ilustración 13 Análisis de territorios basado en los criterios de Boehm y Turner para la empresa SICS

Análisis de los resultados: teniendo en cuenta la tendencia de los criterios de Boehm y Turner en la empresa SICS, se puede inferir que la grafica sugiere adoptar una metodología ágil (ver Ilustración 13), pero basada en métodos más formales o buenas prácticas de gestión para incluir o reforzar tareas orientadas a las personas. Lo anteriormente planteado corrobora la utilización de SCRUM como metodología de desarrollo, pues pondera en territorio ágil conforme a las características de la misma, según lo sugerido por el análisis de los indicadores planteados por Boehm y Turner.

3.1.3 Análisis de las variables dependientes

Análisis de la variable agilidad

La Tabla 4, muestra un resumen de la variable dependiente agilidad. La **dimensión Documentación precisa**, medida en el indicador Cantidad de artefactos, analizó el expediente que se gestionaba anteriormente, adaptada a la metodología RUP y caracterizada por tener un grupo de documentos que robustecían el proceso, duplicando en muchos casos información, donde además se usaban mecanismos robustos pudiéndose aprovechar método ágiles. Se logró disminuir la cantidad de artefactos del expediente a 12, sin perder información relevante del proceso. Como resultado se trabaja con siete artefactos menos respecto al estado original.

En el caso de la ***dimensión Nivel Autogestión del equipo***, medida en el indicador Autonomía del equipo, se estudió la capacidad del equipo para tomar decisiones en el proceso de desarrollo. De las entrevistas realizadas se pudo concluir que tanto antes como después de haberse aplicado el modelo, los equipos de desarrollo contaban con alta independencia y autonomía sobre la toma de decisiones, por lo que este indicador no varía. El otro indicador que mide el Nivel Autogestión del equipo es el auto-enriquecimiento, estudiándose el mismo a partir de los mecanismos de gestión del conocimiento aplicados y asumidos. A partir de las encuestas realizadas se pude concluir que

antes de la aplicación del modelo el auto-enriquecimiento del equipo era bajo y lento, evidenciándose una mejora con el nuevo modelo, a partir de los mecanismos de gestión del conocimiento establecidos (Wiki, Foro, talleres, cursos de superación, otras actividades de capacitación, el propio modelo de desarrollo arquitectónico) aunque las acciones realizadas todavía no cubren todos los requerimientos necesarios para contar con un alto auto-enriquecimiento del equipo.

Tabla 4 Análisis de la variable agilidad

Variable	Dimensión	Indicador	Antes	Después	Variación
Agilidad	Documentación precisa	Cant. de artefactos	19	12	-7
	Nivel Autogestión del equipo	Autonomía	Alta	Alta	Ninguna
		Auto-enriquecimiento	Baja	Media	Mejora

Análisis de la variable reutilización

La Tabla 5, muestra el resumen del análisis realizado de la variable dependiente reutilización. La **dimisión Mecanismos de Reutilización**, medida con el indicador Cantidad de Mecanismos de Reutilización (CMR) establecidos, situada a la derecha de la tabla, muestra como antes de la implantación del modelo no existía ninguno y posterior a la implantación del mismo, se definieron cuatro mecanismos, los cuales son: repositorio de activos, modelo de desarrollo arquitectónico, modelo de desarrollo de software basado en LPS y componentes reutilizables.

Tabla 5 Análisis de la variable reutilización

	Cantidad de Activos Reutilizados						**Cantidad de Mecanismo de Reutilización establecidos**
	Temáticos		Tecnológicos			**Total**	
	LPS1	LPS2	.Net	PHP	Java		
Antes	0	0	0	0	0	0	0
Después	11	11	9	7	0	38	4
Variación	11	11	9	7	0	38	4

Para la **dimensión Activos Reutilizados**, medida en el indicador Cantidad de activos reutilizados, se realiza el análisis a partir de los activos temáticos y tecnológicos. En el caso de los activos temáticos, el análisis se personalizó por las dos LPS creadas, mientras en el caso de los tecnológicos a partir los marcos de trabajo bases que se establecieron. La LPS1 conformó 11 activos luego de la implantación del modelo, obteniendo el mismo resultado la LPS2, donde ambas líneas anteriormente a la implantación del modelo no contaban con ninguno. De la misma forma sucede con los activos tecnológicos, donde antes del modelo no existía ninguno y luego de su aplicación, se crearon nueve activos para el marco de trabajo de .Net y siete para el de PHP, sumando entre todos los activos reutilizados un total de 38.

Análisis de la variable Productividad

La Tabla 6, muestra un resumen de los indicadores Tiempo de terminación del producto y Cantidad de productos, de las dimisiones **Ciclo de vida del producto y Obtención de nuevos**

productos respectivamente, a partir del análisis realizado antes y después de la aplicación del modelo.

Tabla 6 Análisis de los indicadores Tiempo de terminación del producto y Cantidad de productos

Antes de la aplicación del modelo								
Cantidad de productos	**Viajero 1.0**		**Combustible 1.0**		**Inventarios Ociosos**			
Tiempo de terminación del producto	17		11		6			
Ciclo de vida Promedio	11,33333333							
Después de la aplicación del modelo								
Cant. de productos	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**
Tiempo de terminación del producto	7	2	1,7	2,7	2,3	1,9	7	5,6
Ciclo de vida Promedio	3,775							

Analizando la **dimensión Ciclo de vida del producto** antes de la aplicación del modelo se puede concluir que el mismo presentaba un ciclo de vida promedio de 11,33 meses, luego de la aplicación del mismo este tiempo disminuyó a 3,7 meses, evidenciándose una respuesta más rápida a los clientes, lo cual impacta directamente en aumentar la productividad del proceso de desarrollo y por consiguiente de la empresa.

Al analizar la **dimensión Obtención de nuevos productos**, se puede afirmar que antes de la aplicación del modelo se contaba con tres productos desarrollados, significando que este análisis es 26 meses anteriores a la fecha de inicio de aplicación del nuevo modelo, antes de ese período no se cuenta con registro histórico del proceso productivo en la empresa. Luego de la aplicación del modelo el indicador Cantidad de productos aumento a ocho proyectos introducidos, en un periodo de tiempo de 13 meses que duro el experimento, mostrando una variación de cinco productos por encima y usando menos tiempo para su realización.

La **dimensión Eficiencia** se plantea calcular siguiendo la ecuación: $E = \frac{CW}{CP*CV}$

Siendo: E= eficiencia, CP= Cantidad de personas involucradas en el proceso, CV= Ciclo de vida promedio del desarrollo de productos y CW= Cantidad de trabajo, calculada de la siguiente manera: $CW = CProd * IC$

Donde Cprod= la cantidad de productos desarrollados y IC= índice de Complejidad. A su vez el IC= $\sum_1^n (Tnorma * Cantidad\ de\ productos\ según\ complejidad)$. La Tabla 7, muestra la complejidad de los productos de software antes de aplicar el modelo y luego de haberlo aplicado, evidenciando después de la aplicación del modelo, un incremento de complejidad en las soluciones de software.

Tabla 7 Cálculo de la complejidad de solución de productos

		Comp. Baja	**Comp. Media**	**Comp. Alta**	
	TNorma	0,1	0,4	1	Total
Antes	**Cantidad de productos(Cprod)**	0	2	1	3
	Índice de complejidad (IC)	0	0,8	1	1,8

Después	**Cantidad de productos(Cprod)**	1	3	4	8
	Índice de complejidad (IC)	0,1	1,2	4	5,3

La Tabla 8, muestra el cálculo de eficiencia realizado antes y después de la aplicación del modelo, evidenciando un aumento significativo del indicador.

Tabla 8 Cálculo de eficiencia

	CP	**CV**	$\mathbf{CW = CProd * IC}$	**E**	**Eficiencia**
Antes	17	11,33333333	3*1,8	$E=\frac{3*1,8}{17*11,33}$	0,028027682
Después	24	3,775	8*5,3	$E=\frac{8*5,3}{24* 3,775}$	0,46799117

3.2 Caso de estudio DESOFT.

3.2.1 Cubrimiento de los elementos fundamentales que debe contemplar la LPS.

El análisis se realizó tomando en cuenta una implantación experimental del modelo en dominios de aplicación que no involucran desarrollo para hardware específico ni complejos ciclos de aprendizaje. Por esta razón los resultados que se muestran no son concluyentes, representan datos limitados de la implantación piloto realizada. Sin embargo estos resultados permitieron aprobar la generalización del modelo al resto de la institución, la cual se encuentra en proceso de aplicación.

Antes del modelo:

Dominio: el objetivo de la institución es construir soluciones de software para informatizar la sociedad cubana, sin embargo el dominio del las aplicaciones que se desarrollaban no estaba claro, por tanto no se analizaban los procesos de los entornos de negocio objetivos.

Familia de producto: no existía una conceptualización de familias de productos, el proceso productivo estaba orientado a crear una gama abierta y poco organizada en cuanto tipología de producto que iban naciendo según la demanda de clientes, por lo que aún cuando se contaban con 22 proyectos de desarrollo, no se pueden catalogar en su esencia como familias de productos. De los 22 productos existentes solo uno por su dimensión, complejidad técnica e impacto podría asociarse a la gama de sistema de gestión de indicadores y análisis de tendencia, tal es el caso del proyecto Agricultura y precisiones.

Modelo de producción: existía un modelo de producción basado en RUP, con procedimientos de calidad extensos y no funcionales, los desarrollos se realizaban organizados por proyectos de equipos de tres personas. Los mecanismos de seguimiento y control se centraban en el chequeo de la facturación y no en indicadores o procedimientos de seguimiento y control.

Arquitectura: no se conocía la disciplina de la Arquitectura de Software, por tanto no se realizaban estudios arquitectónicos del producto, no se definía la Línea Base, ni se contaba con marcos de trabajo preconcebidos y adaptados al entorno, para el desarrollo del producto.

Activos de software: no se recogía información relativa a los activos de software reutilizables.

<u>Después del Modelo (piloto)</u>:

Dominio: se formalizaron de dos LPS orientadas a los Sistemas de Gestión Administrativa y los Sistemas de Gestión Bancarios.

Familia de producto: Las LPS agruparon el 91% de los proyectos existentes en la empresa. La definición de las mismas se realizó siguiendo estrategias distintas, en el primer caso la estrategia seguida fue reactiva (6), la que consistió en la identificación de los principales productos de la empresa relacionado con la temática, sus tecnologías base, los elementos computacionales más robustos y se realizó una reingeniería sobre los mismos con el objetivo de evolucionar activos núcleos que permitieran iniciar la evolución al modelo LPS. En el segundo caso, la estrategia utilizada fue incremental (6), en la que se partió por realizar un estudio organizacional del segmento de mercado y los procesos negocio asociados a este, consultándose el modelo BIAN (62), desarrollándose en paralelo los activos núcleos tecnológicos y definiéndose las familias de productos a desarrollar. Entre las familias de productos identificadas se conceptualizaron los siguientes:

LPS2- Sistemas de gestión administrativa (soporta el desarrollo en tecnología PHP y .Net (Mono))

- P1-Sistema de facturación Cempalab.
- P2-Sistema de facturación Ensume.
- P3-Sistema de gestión financiera BINSOC (en conceptualización).
- P4-Sistema de gestión administrativa Consultoría Jurídica.
- P5-Sistema de gestión empresa Control Pecuario.
- P6-Sistema de gestión control de casas de rentas. Instituto nacional de la Vivienda.
- P7-Plataforma para la gestión administrativa empresa SICS.

LPS1-Sistemas de gestión bancario (soporta el desarrollo en Tecnología .Java, Grails FrameWork)

- P1-Sistema de gestión bancario Banco Central de Cuba (en conceptualización).
- P2-Sistema gestión bancario BPA (en conceptualización).
- P3-Sistema de gestión bancario Metropolitano (en conceptualización).
- P4-Sistema de gestión bancario para casas financieras (en conceptualización).

Modo de producción: el resultado del modelo piloto permitieron aprobar la extensión del modelo a la totalidad del proceso productivo. El tiempo de implantación se efectuó según lo planificado, constituyendo como principal obstáculo la capacidad de asimilación del nuevo estilo de trabajo, administración y técnicas de ingeniería. Se logró implantar un 95% de los elementos del modelo. Solo fue excluido un único elemento, contemplado en el área de Administración, encargado de la Gestión de Costo. El motivo de su exclusión fue la ausencia de un procedimiento de gestión de costo en el sistema de gestión económica de la institución.

Arquitectura: se homogenizan las soluciones de desarrollo concentrando todo el proceso

productivo sobre tecnología .Net compatibilizado con Mono, PHP y Java, soportándose en una infraestructura tecnológica y de marcos de trabajo desarrollados e implantados con ese objetivo. Se introduce un modelo de desarrollo arquitectónico que central la dirección técnica de las distintas LPS y dirige las estrategias de reutilización e integración de los productos.

Activos de software: se desarrollan e implantan tres marcos de trabajo que pasan a constituir la base tecnológica del proceso productivo. Uno desarrollado con tecnología. .Net y compatibilizado con Mono, otro desarrollado con tecnología PHP basado en el marco de trabajo del ERP Cubano Sauxe y un tercer marco de trabajo basado en Grails Framework de la plataforma Java. Otros activos temáticos producidos, de mayor impacto en los índices de reutilización se encuentran en los (Anexo 54) y (Anexo 55) para las dos LPS referidas.

3.2.2 Análisis de criterios según Boehm y Turner.

Tamaño: DESOFT es una empresa grande, con generalización en todo el país, pero los equipos de trabajo son pequeños, con aproximadamente entre tres y ocho personas. Los niveles de comunicación son sencillos. El conocimiento explícito se gestiona por medio del contacto personal, talleres, correo, gestores documentales, entre otros, siempre no se realiza de forma óptima. Los proyectos son mayormente de duración entre pequeña y mediana. Por tanto se sitúa al criterio en una categoría baja.

Criticidad: no están involucradas pérdidas de vida y la pérdida de recursos es medianamente baja, pues la tendencia apunta a una retención del personal por parte de la empresa. Los ingresos de la empresa y los trabajadores pueden verse afectados cuando no se cumplen los planes de producción, lo que influye directamente en el salario de estimulación de los trabajadores. Con respecto al confort, vale decir que se cuenta con internet limitado solamente a un grupo de trabajadores, no representa una tarea muy compleja obtener las tecnologías necesarias, aunque siempre no están actualizadas. Los espacios de trabajo y los recursos (PC, servidores no alcanzan o quedan en condiciones de hacinamiento, haciendo compleja la concentración en las tareas. Por lo anteriormente expresado situamos este criterio en una escala baja-media.

Dinamismo: los requerimientos en el entorno de desarrollo de software no cambian a menudo, aunque si pueden existir cambios moderados en el entorno clientes. De la misma forma sucede en los contextos técnicos donde los cambios pueden estar dados por la introducción de nuevas tecnologías; incluyendo también los requisito de negocio que requieren cambios en la tecnología. Por tal motivo se ubica en una escala media.

Personal: DESOFT posee predominantemente un personal joven pero con un nivel de preparación aceptable, compuesto en su mayoría por universitarios. También cuenta con personal de basta experiencia en el desarrollo de software, aunque medianamente desactualizado en las tecnologías. Por lo anteriormente planteado se ubica el indicador en una escala media.

Cultura: la organización de la documentación es exigida pero ligera, el control de la gestión de los cronogramas es fuerte, el formalismo de las reuniones y las comunicaciones es ligero, pero coordinado y dirigido por sistemas de trabajo, aunque no tan formales y estables. Las personas se acogen a sistemas de trabajo y tareas claramente definidas en alcance y tiempo, aunque no siempre siguen estándares, procesos y políticas de la forma correcta. Las personas tienen un margen de libertad a la hora de alcanzar los objetivos y no tienen roles definidos. Por tal razón ubicamos a la cultura en una escala entre media y baja.

Análisis de los resultados: teniendo en cuenta la tendencia de los criterios de Boehm y Turner en la empresa DESOFT, se puede inferir que la gráfica sugiere adoptar una metodología ágil (ver Ilustración 14), con necesidad de incluir o reforzar las tareas orientadas a las personas, basándose en métodos más formales o buenas prácticas de gestión. Lo anteriormente planteado corrobora la utilización de SCRUM como metodología, pues pondera en territorio ágil, según lo sugerido por el análisis de los indicadores planteados por Boehm y Turner.

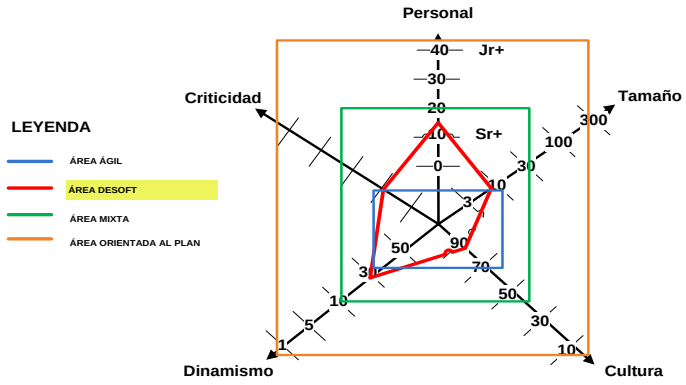


Ilustración 14 Análisis de territorios basado en los criterios de Boehm y Turner para la empresa DESOFT

3.2.3 Análisis de las variables dependientes.

Análisis de la variable agilidad

La Tabla 9, muestra un resumen del comportamiento de la variable dependiente agilidad. La **dimensión Documentación precisa**, medida en el indicador Cantidad de artefactos, analizó el expediente que se gestionaba anteriormente, caracterizado por ser adaptado a la metodología RUP y tener un grupo de documentos que robustecían el proceso, donde muchos de ellos en ocasiones ni se usaban, lo que conllevaba a procedimientos de calidad extensos y no funcionales. Se logró disminuir la cantidad de artefactos del expediente a 12, sin perder información relevante del proceso. Como resultado se trabaja con 11 artefactos menos respecto al expediente original.

En el caso de la ***dimensión Nivel Autogestión del equipo***, medida en el indicador Autonomía del equipo, se estudió la capacidad del equipo para tomar decisiones en el proceso de desarrollo. De las entrevistas realizadas se pudo concluir que tanto antes como después de haberse aplicado el

modelo, los equipos de desarrollo contaban con alta independencia y autonomía sobre la toma de decisiones en el proceso productivo, por lo que este indicador no varía. El otro indicador que mide el Nivel Autogestión del equipo es el Auto-enriquecimiento, en el que se puede concluir que antes de la aplicación del modelo el Auto-enriquecimiento prácticamente no se gestionaba, ponderando el mismo en una escala baja. Sin embargo después de la aplicación de este, se evidencia una mejora, a partir de los mecanismos de gestión del conocimiento establecidos (Wiki, Foro, talleres, cursos de superación, otras actividades de capacitación, el propio modelo de desarrollo arquitectónico), aunque se plantean la necesidad de realizar un trabajo más profundo en este sentido.

Tabla 9 Análisis de la variable agilidad DESOFT

Variable	Dimensión	Indicador	Antes	Después	Variación
Agilidad	Documentación precisa	Cantidad de artefactos	23	12	-11
	Nivel Autogestión del equipo	Autonomía	Alta	Alta	Ninguna
		Auto-enriquecimiento	Baja	Media	Mejora

Análisis de la variable Reutilización

La Tabla 10, muestra el resumen del análisis realizado de la variable dependiente Reutilización. La **dimisión Mecanismos de Reutilización**, medida con el indicador Cantidad de Mecanismos de Reutilización establecidos, situada a la derecha de la tabla, muestra como antes de la implantación del modelo no existía ninguno y posterior a la implantación del mismo, se definieron cuatro mecanismos, los cuales son: repositorio de activos, modelo de desarrollo arquitectónico, modelo de desarrollo de software basado en LPS y componentes reutilizables.

Tabla 10 Análisis de la variable reutilización DESOFT

	Cantidad de Activos Reutilizados						**Cantidad de Mecanismo de Reutilización establecidos**
	Temáticos		Tecnológicos			**Total**	
	LPS1	LPS2	.Net	PHP	Java		
Antes	0	0	0	0	0	0	0
Después	5	9	9	11	9	43	4
Variación	5	9	9	11	9	43	4

Para la **dimensión Activos Reutilizados**, medida en el indicador Cantidad de activos reutilizados, se realiza el análisis a partir de los activos temáticos y tecnológicos. En el caso de los activos temáticos, el análisis se personalizó por las dos LPS creadas, y en el caso de los activos tecnológicos, se realizó a partir de los marcos de trabajo bases que se establecieron. De la LPS Banco se lograron conformar cinco activos para la fase de desarrollo en que se encuentra esta. Por otro lado en la LPS Gestión Administrativa, se lograron conformar nueve activos luego de la implantación del modelo, donde anteriormente a la implantación no existían activos reutilizables, al no utilizarse el paradigma LPS. En el caso de los activos tecnológicos, antes de la implantación del modelo no existía ninguno y luego de su aplicación, se crearon nueve activos para el marco de

trabajo de .Net, once para el de PHP y nueve para el de Java, sumando entre todos un total de 43.

Análisis de la variable Productividad

La Tabla 11, muestra un resumen de los indicadores Tiempo de terminación del producto y Cantidad de productos, de las dimisiones Ciclo de vida del producto y Obtención de nuevos productos respectivamente, a partir del análisis realizado antes y después de la aplicación del modelo.

Tabla 11 Análisis de los indicadores Tiempo de terminación del producto y Cantidad de productos DESOFT

Antes de la aplicación del modelo						
Cantidad de productos	22 Productos					
Ciclo de vida Promedio	9,7					
Después de la aplicación del modelo						
Cantidad de productos	P1	P2	P3	P4	P5	P6
Tiempo de terminación del producto	2	5	2	2	2	2
Ciclo de vida Promedio	2,5					

Analizando la **dimensión Ciclo de vida del producto** antes de la aplicación del modelo se puede observar que se contaban con 22 productos, siendo de ellos el más significativo el proyecto Agricultura y precisiones. Todos estos proyectos sugieren un ciclo de vida promedio de 9,7 meses. Luego de la aplicación del modelo se crearon seis nuevos productos en 2,5 meses como promedio, evidenciándose una disminución en el tiempo de terminación de los productos, lo que propicia una respuesta más rápida a los clientes, impactando directamente en aumentar la productividad del proceso de desarrollo y por consiguiente de la empresa.

Al analizar la **dimensión Obtención de nuevos productos**, se puede afirmar que antes de la aplicación del modelo se contaba con 22 productos desarrollados, significando que este análisis es 18 meses anteriores a la fecha de inicio de la aplicación del nuevo modelo, antes de ese período no se cuenta con registro histórico del proceso productivo en la empresa. Luego de la aplicación del modelo se introdujeron seis productos, en un período de tiempo de cinco meses que duró el experimento. El indicador no evidencia un aumento en la cantidad de productos, pero es necesario tener en cuenta que los resultados después de la aplicación del modelo se refieren a un piloto con un tiempo de duración mucho menor que el estudiado antes de la aplicación del mismo. Para que el resultado de este indicador sea concluyente, es necesario seguir observando su tendencia después de la generalización del modelo en toda la empresa, y tomando en consideración el mismo período de tiempo observado antes de su aplicación y las cantidades de recursos humanos involucrados, en el caso del piloto se contó solamente con nueve especialistas.

La **dimensión Eficiencia** se plantea calcular siguiendo la ecuación: $E = \frac{CW}{CP*CV}$

Siendo: E= eficiencia, CP= Cantidad de personas involucradas en el proceso, CV= Ciclo de vida promedio del desarrollo de productos y CW= Cantidad de trabajo, calculada de la siguiente manera: $CW = CProd * IC$. Donde Cprod= la cantidad de productos desarrollados y IC= índice de

Complejidad. A su vez el IC= $\sum_1^n(Tnorma * Cantidad\ de\ productos\ según\ complejidad)$. La Tabla 12, muestra la complejidad de los productos de software antes de aplicar el modelo y luego de haberlo aplicado, pudiendo concluir que la complejidad de los productos antes del experimento era mayor que la de los seleccionados para el experimento.

Tabla 12 Cálculo de la complejidad de solución de productos DESOFT

		Comp. Baja	Comp. Media	Comp. Alta	
	TNorma	0,1	0,4	1	Total
Antes	**Cantidad de productos(Cprod)**	9	11	2	22
	Índice de complejidad (IC)	0,9	4,4	2	7,3
Después	**Cantidad de productos(Cprod)**	4	2	0	6
	Índice de complejidad (IC)	0,4	0,8	0	1,2

La Tabla 13, muestra el cálculo de eficiencia realizado antes y después de la aplicación del modelo, evidenciando un aumento de la eficiencia, pero teniendo en cuenta que la implantación del modelo no estaba generalizada, por tanto no incluía al 100% de los recursos humanos, por lo que se considera un resultado parcial.

Tabla 13 Cálculo de eficiencia DESOFT

	CP	CV	$\mathbf{CW = CProd * IC}$	E	Eficiencia
Antes	87	9,7	22*7,3	$E=\frac{22*7,8}{87*9,7}$	0,192519779
Después	9	2,5	6*1,2	$E=\frac{6*1,2}{9*2,5}$	0,32

3.3 Análisis de la variable independiente Calidad del modelo.

Entre los indicadores que persigue medir la Calidad del modelo está **la Calidad de la documentación**, en este sentido en ambas empresas existía un expediente adaptado a la metodología RUP que complejizaba y robustecía la gestión de la documentación. Se logró adaptar el expediente al modelo de LPS propuesto y a la metodología ágil de desarrollo SCRUM (más a fin con estos entornos), reduciendo considerablemente el número de artefactos, sin perder información, como muestra Ilustración 15. El modelo además realiza la descripción de todas las actividades de los procesos señalando los artefactos que se instancian en estas actividades. Por otra parte como resultado del trabajo se construyó un expediente documental que contiene la especificación de cada artefacto referido en la explicación de los procesos (ver epígrafe 2.3.6). La calidad del modelo antes de su aplicación es valorada como poco adecuada y luego de su aplicación se cataloga como adecuada, evidenciando resultados positivos respecto a esta variable.

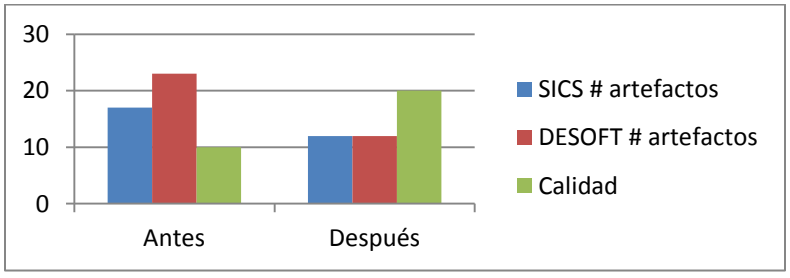


Ilustración 15 Reducción del número de artefactos

En el caso del análisis del indicador **Capacidad de ser generalizado**, se evaluó en una escala del cero al cinco el nivel de implantación de los procesos, estableciendo el nivel de significancia según orden de numeración. Del análisis se puede concluir que se logró la implantación del modelo en la empresa SICS a partir de dos LPS, excluyendo el área de Administración de la LPS, lo que representa 9 procesos implantados de los 14 que el modelo propone. Mientras en la empresa DESOFT se implantó el modelo parcialmente mediante un piloto a partir de dos LPS, excluyendo la Gestión de Costos perteneciente al área de Administración de la LPS, lo que representa 13 proceso implantados, conforme muestra Ilustración 16.

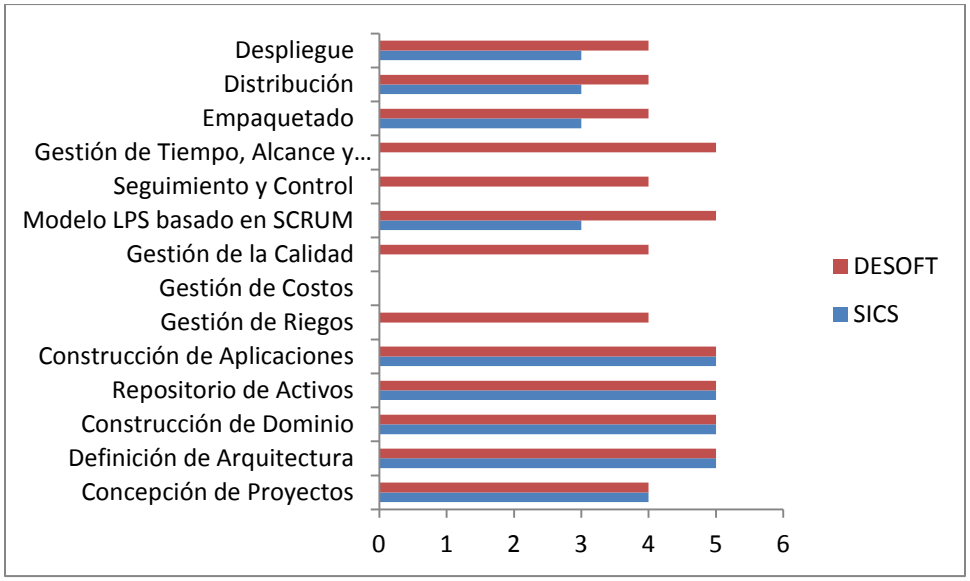


Ilustración 16 Proceso del modelo aplicados a las empresas SICS y DESOFT

La Ilustración 17 muestra que la tendencia hacia la cantidad de procesos implantados crece para la segunda generalización del modelo, lo que evidencia maduración en este y su proceso de implantación. Se puede concluir que se establecieron en ambas empresas dominios de solución, familias de productos, activos reutilizables, modo de producción y arquitecturas tipo. Por esta razón se cataloga el indicador como alto.

Otro de los indicadores es la **Completitud**, analizado desde la perspectiva del nivel en que el modelo propuesto es capaz de cubrir las áreas prácticas del modelo perteneciente al SEI. El Anexo 61, muestra los por cientos de cubrimiento de las áreas prácticas del modelo canónico del SEI (5).

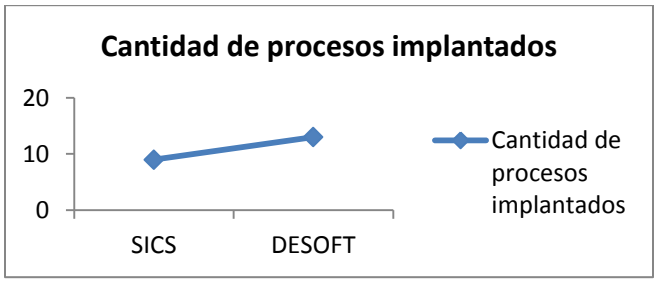


Ilustración 17 Tendencia de la cantidad de procesos implantados del modelo en las empresas SICS y DESOFT

Del análisis de los indicadores del Anexo 61 se puede concluir que de las 29 áreas propuestas por el SEI, nueve no se cubren en ninguna medida, siete se cubren parcialmente con altos porcientos, cuatro con bajos por cientos y nueve se cubren totalmente, representando un total de 20 áreas instanciadas de alguna manera, conforme muestra Ilustración 18. Por esta razón se le atribuye al indicador Completitud un valor adecuado.

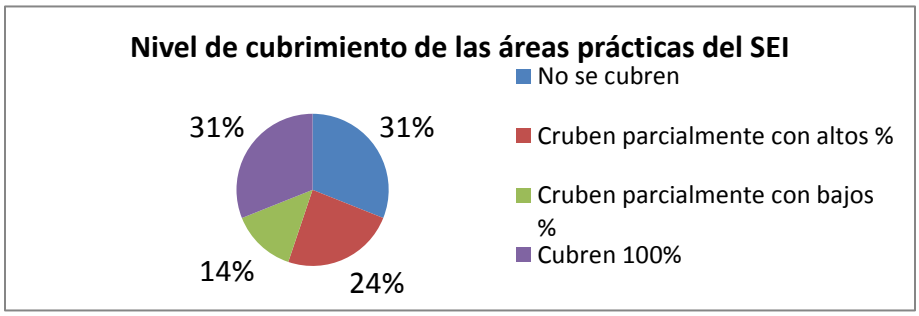


Ilustración 18 Nivel de cubrimiento de las áreas prácticas del modelo del SEI

Conclusiones parciales

- Se desarrolló la implantación del modelo en la empresa SICS, excluyendo el área de Administración de la LPS. Mientras en la empresa DESOFT se implantó el modelo parcialmente mediante un piloto, excluyendo la Gestión de Costos perteneciente al área de Administración de la LPS. Se establecieron en ambas empresas dominios de solución, familias de productos, activos reutilizables, modo de producción y arquitecturas tipo.
- Se evaluaron en cada una de las empresas el comportamiento de las variables agilidad, reutilización y eficiencia, antes y después de la aplicación del modelo, mostrando resultados positivos en ambos casos.
- El tipo de metodología de desarrollo de software que más se adecúa a los entornos de producción de las empresas SICS y DESOFT es el enfoque ágil, con necesidad de incluir o reforzar las tareas orientadas a las personas, basándose en métodos mas formales o buenas prácticas de gestión. De esta manera se corrobora la utilización de SCRUM como metodología de desarrollo factible a usarse en estas empresas.
- El modelo propuesto cubre gran parte de las áreas prácticas del modelo tomado como base pertenécete al SEI y además cuenta con una buena capacidad de ser generalizado, por lo que se puede concluir de que las metas de calidad establecidas para el mismo fueron alcanzadas.

CONCLUSIONES GENERALES

- La evolución de los modelos de procesos de desarrollo de software ha dado lugar a un nuevo paradigma que rompe la barrera del desarrollo de software tradicional, denominado Modelo basado en Líneas de producción de Software (LPS). Entre los modelo de LPS más estudiados se encuentra el propuesto por el SEI, constituyendo este punto de partida para la presente investigación. Las metodologías ágiles prometen ser las que más se ajustan al entorno cubano de desarrollo de software, siendo SCRUM una de estas metodologías agiles más usadas a nivel internacional y constituyendo esta, una alternativa aprovechable para integrar al modelo de LPS y conformar un proceso de desarrollo de software que estandarice y agilice la producción de software para contribuir al desarrollo de la sociedad cubana y el aumento de la productividad de sus empresas.
- El modelo aporta como principal resultado la vinculación de técnicas de desarrollo ágiles en las áreas de administración de un modelo industrial. Otro aporte significativo del trabajo radica en la introducción de un modelo de desarrollo arquitectónico, que defina el mapa de domino y por consiguiente, el plan de producción de las diferentes áreas productivas contempladas en el modelo inicial LPS del SEI.
- El modelo agrupa cuatro áreas fundamentales: Administración, Arquitectura, Ingeniería de Dominios e Ingeniería de Aplicaciones, todas sinérgicamente relacionadas, que garantizan niveles de madurez e integración para la administración y desarrollo de un proceso productivo. Propone además cinco fases fundamentales: Concepción, Conceptualización Arquitectónica, Ingeniería de Dominios, Ingeniería de Aplicaciones y Empaquetado y Despliegue, todas ellas gestionadas por la metodología de desarrollo SCRUM y los principales postulados de gestión de proyectos.
- El modelo propone la guía de actividades que describen los procesos ingenieriles y de gestión, los roles involucrados en estos procesos y los artefactos y documentos que se centralizan en un expediente digital organizado.
- El modelo ha sido probado en dos entornos experimentales con características diferentes excluyendo los dominios que no involucran desarrollo para hardware específico ni complejos ciclos de aprendizaje en las empresas. Los resultados obtenidos validan beneficios económicos de eficiencia y adaptabilidad del proceso.
- Se considera que los objetivos propuestos fueron cumplidos, ya que se propone un modelo basado en LPS y técnicas de desarrollo ágil, introduciendo agilidad en las empresas SICS y DESOFT, expresada a en elevar el nivel de autogestión del equipo y reducir la documentación sin pérdida de información. Por otro lado el modelo introduce reutilización y productividad, los que contribuye a disminuir tiempo y esfuerzo, acortando el ciclo de vida de desarrollo y por tanto incrementando la capacidad de obtención de nuevos productos. El modelo propuesto es de calidad aceptable, cubre con gran parte de las áreas de conocimiento del modelo del SEI, cuenta con capacidad de ser generalizado y propone un expediente documental que da soporte a todos los procesos.

RECOMENDACIONES

- Proponer al modelo herramientas que den soporte a todo el proceso descrito desde la perspectiva de gestión de proyecto, los procesos ingenieriles, hasta la gestión de repositorios.
- Completar la fase de Administración de la LPS con los elementos de Gestión Organizacional planteados por el SEI.
- Se considera que el modelo debe ser monitorizado durante un año más de explotación, evaluándose la tendencia de los indicadores observados hasta el momento.
- Incluir en el modelo implantado en la empresa SICS los elementos que no pudieron ser generalizados pertenecientes al área de Administración: Seguimiento y control, Riesgos, Costos y Calidad. En el caso de la empresa DESOFT incluir el único elemento no implantado referente a la gestión de Costos.
- Ampliar el modelo a dominios de aplicación que involucren desarrollo para hardware específico y complejos ciclos de aprendizaje.
- Incluir al modelo actividades que contribuyan en incluir o reforzar las tareas orientadas a las personas y al desarrollo de los equipos de trabajo, para logar un nivel técnico superior.

BIBLIOGRAFÍA

1. ***Castillo, José Luis Duarte.*** *Factores determinantes y críticos en empresas de servicios para la obtención de ventajas competitivas sostenibles y transferibles a estrategias de globalización:un análisis de la industria del software. Director: Llonch Andreu, Joan. Tesis doctoral. Universdad Autònoma de Barcelona. Departamento de Economía de Empresa, Bellaterra, España, s.n: 8468922870, noviembre de 2005.*

2. ***González, Pilar Rodríguez.*** *SOFTWARE, ESTUDIO DE LA APLICACIÓN DE METODOLOGÍAS ÁGILES PARA LA EVOLUCIÓN DE PRODUCTOS. Director: Garbajosa Sopeña, Juan. Tesis mestría. FACULTAD DE INFORMÁTICA UNIVERSIDAD POLITÉCNICA DE MADRID, Madrid, España, septiembre de 2008.*

3. ***Hernández, Vismar Santos.*** *Estudio sobre la industria del Software a nivel mundial. Caracterización en América Latina y Cuba. Observatorio de la Economía Latinoamericana, eumed.net, Nº 116, 2009, p. 3-23.*

4. ***Bend, Ken.*** *Manifiesto por el Desarrollo Ágil de Software. Manifiesto por el Desarrollo Ágil de Software. [En línea] 2001. [Citado el: 15 de 6 de 2011.] http://www.agilemanifesto.org/iso/es/.*

5. ***Northrop, Linda.*** *Software Engineering Institute (SEI). SEI. [En línea] Universidad de Canie Melou. EE.UU 2008. [Citado el: 15 de 6 de 2011.] http://www.sei.cmu.edu/productlines/framework.html.*

6. ***NORTHROP, Linda.*** *Army Software Product Line Workshop. [En línea] Software Engineering Institute, Carnegie Mellon University, EE.UU [Citado el: 1 de 6 de 2011.] 2006. Disponible en http://www.sei.cmu.edu/productlines/framework.html. s.n: 15213.*

7. ***Torre, Dr. Ing. Angel Notario de la.*** *APUNTES PARA UN COMPENDIO SOBRE METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA. APUNTES PARA UN COMPENDIO SOBRE METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA. Pinar del Río, 1999. ISBN: 85-7919-274-0.*

8. ***LEÓN, ROLANDO ALFREDO HERNÁNDEZ.*** *El paradigma cuantitativo de la investigación científica. La Habana: EDUNIV Editorial Universitaria, 2002. ISBN: 959-16-0343-6.*

9. ***Sommerville, Iam.*** *Ingeniería de Software, 7ta edición. Madrid(España) : Pearson Educación, 2005. ISBN: 84-7929-074-5.*

10. ***Arias, José R. Álvarez y Manuel.*** *Análisis, Diseño y Mantenimiento del Software. MADRID : Dpto. de Inteligencia Artificial - ETSI Informática - UNED, 2002,2007.*

11. ***Pressman, Roger S.*** *Ingeniería de Software: Un enfoque práctico, 6ta edición. EEUU: Pearson Educación, 2005. ISBN: 0072853182.*

12. ***Joan Chipia, Fabiola Pabon y Linda Palencia.*** *Modelo basado en prototipos. Merida : Universidad de los Andes, febrero 2010.*

13. ***Boehm, Barry W.*** *A spital model of software development and enhancement . volumen 21, EEUU : Defense Systems Group, IEEE Computer, mayo 1988. s.n: 0018-88-0500-0061*

14. ***Lidia Fuentes, José M. Troya y Antonio Vallecillo.*** *Desarrollo de Software Basado en Componentes. Málaga, Spain : Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga. ETSI Informática. Campus Teatinos, 2006 .s.n. 29071.*

15. ***Iribarne, Luis.*** *UN MODELO DE MEDIACIÓN PARA EL DESARROLLO DE SOFTWARE BASADO EN*

COMPONENTES COTS. Director:Dr. Antonio Vallecillo Moreno, Dr. Josée María Troya Linero. Tesis doctoral. Departamento de Lenguajes y Computación, Universidad de Almería, España , s.n: 04210, junio 2003.

16. ***Montilva, Jonás A.*** *Desarrollo de Software Basado en Componentes. En: XIII Asamblea General del ISTEC Santa Cruz, Bolivia, Diciembre 2003.*

17. ***Mendoza, Gonzalo Mena.*** *RAD: Desarrollo Rápido de Aplicaciones. [En línea]. Santiago de Querétaro : Universidad Autónoma de Querétaro, [Citado el: 15 de 5 de 2011.] 2005. Disponible en: http://www.mena.com.mx/gonzalo/maestria/ingsoft/presenta/rad/*

18. ***Jack Greenfield, Keith Short, Steve Cook and Stuart Kent.*** *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, 1era edición. EE.UU: Wiley Publising, 2004. ISBN: 0471202843*

19. ***Javier Muñoz, Vicente Pelechano.*** *MDA vs Factorías de Software. Actas del II Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM 2005) Valencia, España : Universidad Politécnica de Valencia, 2006. s.n: 46022.*

20. ***C. Li, H. Li, M. Li,*** *A Software Factory Model Based on ISO9000 and CMM for Chinese Small Organizations. En: Second Asia-Pacific Conference on Quality Software (APAQS'01), Hong Kong, 2001. ISBN: 0-7695-1287-9.*

21. ***Marbys Marante Valdivia, Yanosky Rios La Hoz.*** *Evaluación crítica sobre modelos de factoría de software. [En línea] Habana, Cuba ,[Citado el: 1 de 6 de 2011.] 2008. Disponible en: http://geothesis.com/index.php?option=com_content&view=article&id=586:evaluacirica-sobre-modelos-de-factor-de-software&catid=21:artulos&Itemid=100*

22. ***FERNSTROM, C., NARFELT, K. H y OHLSSON, L.*** *Software Factory Principles, Architecture and Experiments. EE:UU : IEEE Software. (Vol. 9, No. 2), March/April 1999. pp. 36-44. ISBN: 10.1109/52.120600*

23. ***Fernandes, Aguinaldo Aragon y Teixeira, Descartes de Souza.*** *Fábrica de Software:Implementación y Gestión de Operaciones. Editora Atlas, 2004. ISBN: ISBN: 8522436908*

24. ***Basili, Victor. R., Caldiera, G Gianluigi y Cantone, Giovanni.*** *A Reference Archiecture for the Component Factory. ACM Transaction on Software Engineering and Methodology (TOSEM), EE.UU, vol. 1(1) enero 1992. pp 53-80. ISBN: 10.1145/125489.122823*

25. ***ONTIVERO.*** *Software Factories (Parte 2). [En línea] 2008. Disponiblre en: http://software-factories.blogspot.com/2007/09/software-factories-introduccin-parte-2.html.*

26. ***Clements, Paul., Northrop, Linda.*** *Software Product Lines. Practices and Patterns. Boston, EE.UU : Addison-Wesley, 2002, ISBN: 0-201-70332-7*

27. ***Krueger, Charles.*** *Introduction to Software Product Lines. [En línea] [Citado el: 1 de 6 de 2011.] 2006. Disponible en:www.softwareproductlines.com.*

28. ***Gomma, Hassan.*** *Designing Software Product Lines with UML: Fom use cases to patters based softwares arquitectures. Redwood, EE.UU: Addison Wesley, 2004, ISBN: 0201775956*

29. ***Software Product Lines*** *. [En línea] [Citado el: 10 de 6 de 2011.] Carnegie Mellon University, 2009. Disponible en: http://www.sei.cmu.edu/productlines/frame_report/index.html.*

Montilva, Jonás. *Desarrollo de Software Basado en Líneas de Productos de Software. En: Conferencia DVP*

IEEE Computer Society Región 9, 17 de octubre 2006, Disponible en: http://www.ieee.org.ar/downloads/2006-montilva-productos-prt.pdf

30. ***Ochoa, Ing. René Lazo.*** *Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión.Director:Dr. Pedro Piñero, Tesis maestría. Habana, Cuba : UCI Universidad de la Ciencias Informáticas, 2011.*

31. ***Dr.C Pedro Y. Piñero Pérez, MsC Maikel Yelandi Leyva Vázquez.*** *Plan de gestión proyecto vista desde la gestión de integración. Cuba : Maestría de gestión de proyectos, Universidad de las Ciencias informáticas, 2008.*

32. ***Sametinger, Johannes.*** *Software Engineering With Reusable Components. Berlin and New York : 1ra edition, 2002. ISBN: 9783540626954*

33. ***Díaz, Óscar.*** *LÍNEAS DE PRODUCTO SOFTWARE. País Vasco : Facultad de Informática, Universidad del País Vasco, 2010. Disponible en http://alarcos.inf-cr.uclm.es/per/fruiz/cur/santander/odiaz-lineasproducto.pdf*

34. ***Chicote, Cristina Vicente.*** *Desarrollo integral de sistemas de procesamiento de información visual: un enfoque multiparadigma basado en LPS y generación automática de software. Director: Dr. José Carlos Fernandez, Dr.Predro Sanchez. Tesis Doctoral. España : Universidad Politecnica de Cartagena, 2005.*

35. ***Fraunhofer iese****. [En línea] [Citado el: 8 de 6 de 2011.] 2001. Instituto de Experiencia en Ingeniería de software, Munich, Alemania. Disponible en http://www.iese.fraunhofer.de/de/.*

36. ***Colin Atkinson, Joachim Bayer, and Dirk Muthig.*** *Component-Based Product Line Development:The KobrA Approach. Alemania :Fraunhofer Institute for Experimental Software Engineering (IESE), 2007.Disponible en: http://citeseerx.ist.psu.edu/viewdoc/*

37. ***Pino, Ing. Henrik Pestano.*** *Propuesta de modelo de desarrollo para líneas de productos de software en centros de producción. Director: Dr.C Pedro Y. Piñero Pérez, Tesis maestría, CIUDAD DE LA HABANA : Universidad de las Ciencias informáticas, 2011.*

38. ***Torío, Julio Mellado.*** *Estrategia de pruebas de líneas de producto de sistemas de tiempo real especificados con diagramas de estados jerarquicos.Director:dr. Juan Carlos Dueñas López, Tesis doctoral. España : Escuela técnica superior de ingenieros de telecomunicaciones, Departamento de Ingeniería de Sistemas Telemáticos, 2004.*

39. ***Jacobson, Ivar, Booch, Grady y Rumbaugh, James.*** *El proceso unificado de desarrollo de software. Madrid : Pearson Educación, 2000. ISBN: 8478290362.*

40. ***Reynoso, Carlos.*** *Métodos Heterodoxos en Desarrollo de Software. [En línea] [Citado el: 8 de 6 de 2011.] Buenos Aires : UNIVERSIDAD DE BUENOS AIRES, Abril de 2004. Disponible en http://carlosreynoso.com.ar/archivos/arquitectura/Metodos-Agiles.PDF*

41. ***Inteco, Laboratorio Nacional de Calidad del Software de.*** *Curso de desarrollo agil. España : Inteco (Instituto Nacional de Tecnologías de la Comunicación), 2009. Disponible en: http://emprendecaminos.com/docs/%5BAgil%5D-Curso%20de%20Desarrollo%20%C3%81gil.pdf*

42. ***Sanchez, María A. Mendoza.*** *Metodologías De Desarrollo De Software. Perú :2004.Disponible en http://www.willidev.net/InsiteCreation/v1.0/descargas/cualmetodología.pdf.*

43. ***Escribano, Gerardo Fernández.*** *Introducción a Extreme Programming. España : Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos, [En línea] 2002. [Citado el: 13 de 6 de 2011.] Disponible en: http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf*

44. Sito oficial de la Comunidad: Crystal Methodologies. [En línea] 23 de 5 de 2008. [Citado el: 13 de 6 de 2011.] Disponible en: http://crystalmethodologies.blogspot.com/.

45. ***Enrich, Margarita Fernández.*** *Crystal Methodologies. [En línea], 2003 [Citado el: 13 de 6 de 2011.] Valencia : Facultad de Informática, Universidad Politécnica de Valencia, Laboratorio de Sistemas de Información. Disponible www.dsic.upv.es/asignaturas/**facultad**/lsi/trabajos/282002.ppt*

46. ***Paredes, Roberto José Silva.*** *Metodología MSF. definición, características y modelos. [En línea] 2010. [Citado el: 13 de 6 de 2011.] España: Universidad Estatal de Milagro. Disponible en: http://www.slideshare.net/bebeyom/metodologia-msf-4861508.*

47. ***Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera.*** *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. [En línea] 2009. [Citado el: 13 de 6 de 2011.] España : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación. Disponible en adonisnet.files.wordpress.com/.../articulo-**metodologia**-de-sw-formato.doc*

48. ***Deemer, Pete; Gabrielle Benefield; Craig Larman; Bas Vodde.*** *INFORMACIÓN BÁSICA DE SCRUM. [En línea] 2010. [Citado el: 13 de 6 de 2011.] EE.UU :SCRUM Training Institute, 2009. Traducción de Leo Antoli. www.ScrumTI.com. Disponible assets.**scrum**foundation.com/downloads/3/**scrum**primer_es.pdf?1285932063*

49. ***Palacio, Juan.*** *Flexibilidad con SCRUM. 2008. Libro gratuito: Open Educational Resource. Disponible en www.navegapolis.net/content/view/694/*

50. Comunidad de desarrollo y estudio de SCRUM: Proyectos agiles.org. autor principal: Xavier Albaladejo [En línea] 16 de 3 de 2011. [Citado el: 25 de 5 de 2011.] Disponible en http://www.proyectosagiles.org/.

51. ***Gabardini, Juan y Campos, Lucas.*** *Balanceo de metodologías orientadas al plan y agiles.Herramienta para la selección y adaptación. [En línea] 2004. [Citado el: 13 de 6 de 2011.] Buenos Aires, Argentina : Global Congress Proceedings, Universidad Católica de Maule, Escuela de Ingenieía Civil Informática Disponible en :http://www.eici.ucm.cl/Academicos/ygomez/descargas/Ing_Sw2/apuntes/Complementario%20al%20doc%20en%20ingles%20Using%20Risk.pdf*

52. ***Barry Boehm, Richard Turner.*** *Balancing Agility and Discipline: A Guide for the Perplexed. EE:UU : Addison Wesley, 2003. ISBN: 0-321-18612-5.*

53. ***PMI.*** *Fundamentos de la Dirección de Proyectos, Tercera Edición. EEUU : Project Management Institute, Inc, 2004. ISBN:1-930699-73-5.*

54. ***Andrés Romero, Fabián Ceballos.*** *Líneas de Productos de Software Dirigidas por Modelos (MD-SPL): Oportunidades y Retos. Revista: Paradigma en el desarrollo de Software Bogotá, [En línea] 2009. [Citado el: 13 de 6 de 2011.] Colombia : Universidad de los Andes. s.n:18A 10. Disponible en: http://paradigma.uniandes.edu.co/media/articulos/a-romero-4.pdf*

55. ***Lage, Cesar.*** *Arquitectura ERP cubano, marco de trabajo CEDRUS, aquitectura de sistema.Director:MsC: Yadenis Piñero, Tesis mestría, La Habana, Cuba : Universidad de las Ciencias Informáticas, 2011.*

56. ***Ken Schwaber, Jeff Sutherland.*** *Guía sobre Scrum. [En línea] 2010. [Citado el: 13 de 6 de 2011.]. Disponible en http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20ES.pdf#view=fit*

57. ***Perez, MsC. Yadenis Piñero.*** *Curso de Gestión de Riesgos tema 1.Maestría de Gestión de Proyectos Informáticos Habana, Cuba : Universidad de las Ciencias Informáticas, 2009.*

58. ***Alvarez, Miguel Angel.*** *Return On Investment. DesarrolloWeb.com. [En línea] 9 de marzo de 2009. [Citado el: 3 de 6 de 2011.]. Madrid, Disponible en http://www.desarrolloweb.com/articulos/que-es-roi.html.*

59. ***Martin Fowler****.Continuous Integration. [En línea] 1 de 5 de 2006. [Citado el: 2011 de 6 de 1.] Disponible en: http://martinfowler.com/articles/continuousIntegration.html.*

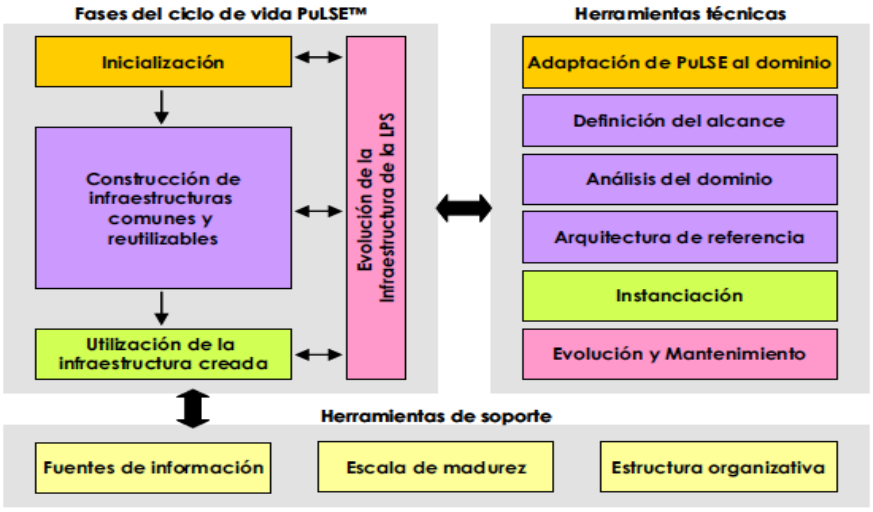
60. ***Foundation, BIAN WG Architecture Framework &.*** *A BIAN Architecture Document. BIAN Metamodel – Explanatory Notes. EEUU, 1 julio 2009.*

61. ***Smith, John.*** *A Comparison of RUP® and XP . EE.UU : Rational Software White Paper, 2001, n.s: 95014.*

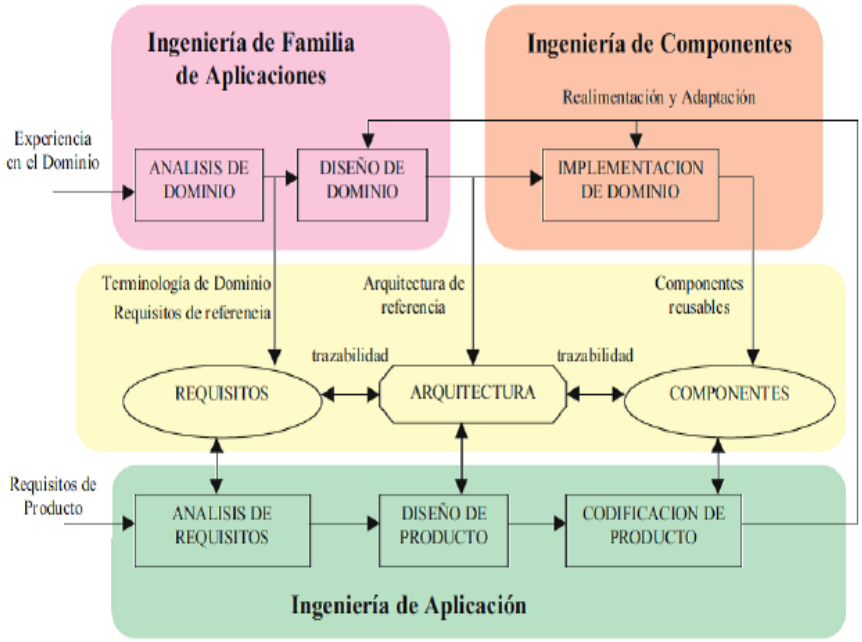
62. ***Gimenez, Carlos.*** *Costos para Empresarios. Caja de herramientas. [En línea] 2009. [Citado el: 2011 de 6 de 1. Dsiponible en: http://www.abcpymes.com/menu22.htm.*

ANEXOS

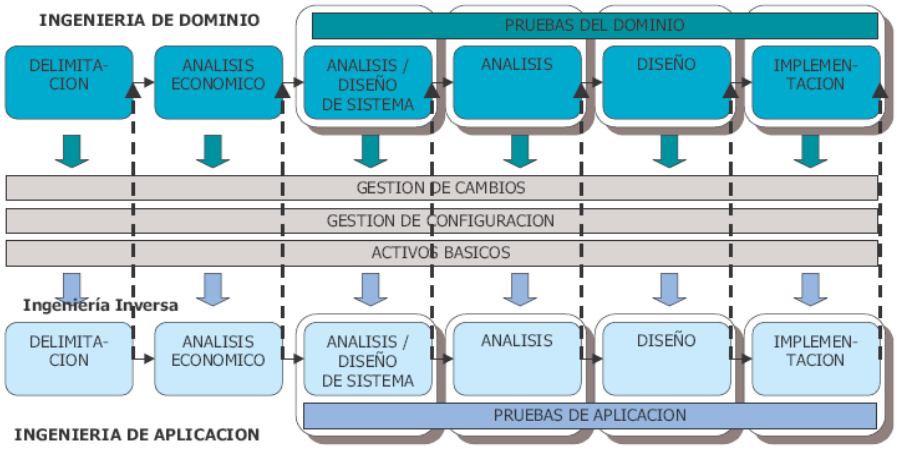
Anexo 1 Vista general de PuISE (40)



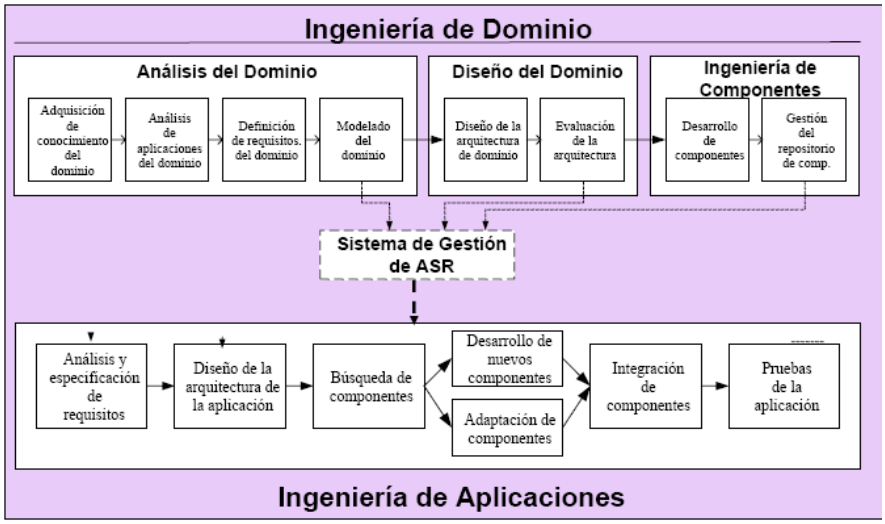
Anexo 2 Proceso de desarrollo PRAISE (40)



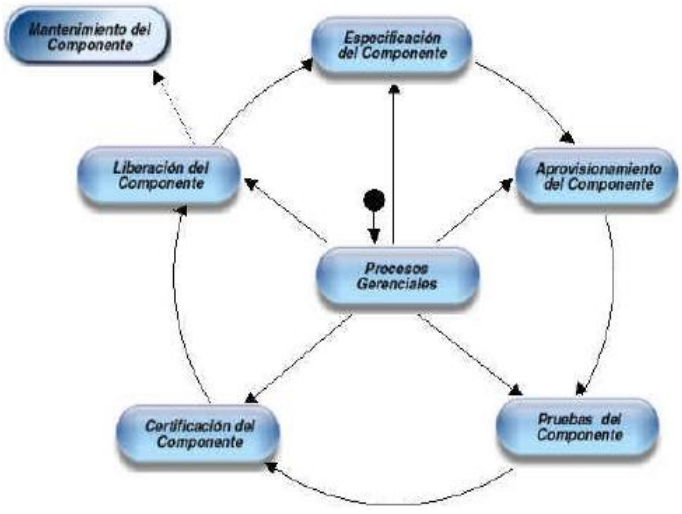
Anexo 3 Modelo de referencia CAFE (40)



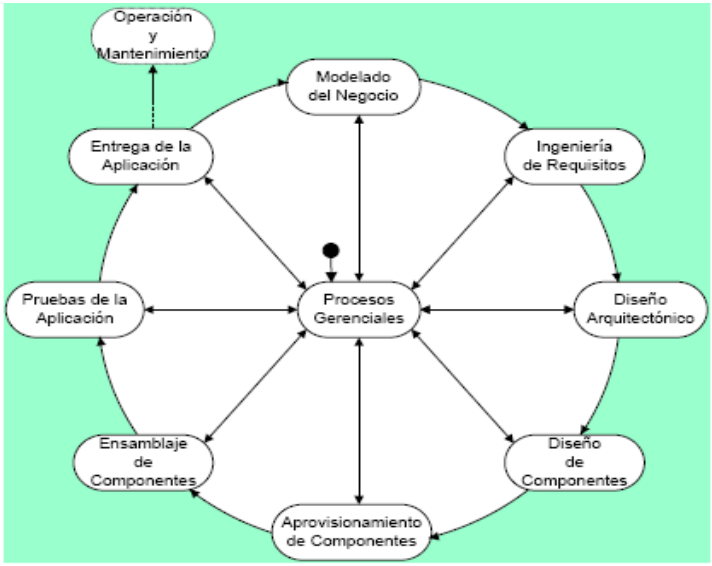
Anexo 4 Proceso del modelo TWIN (33)



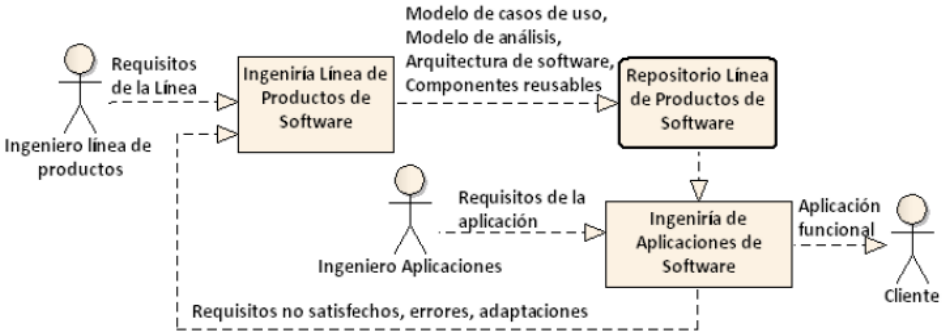
Anexo 5 Modelo de procesos del Método WATCH-Component (33)



Anexo 6 Modelo de procesos de Metodo WATCH- Application (33)



Anexo 7 Modelo de procesos SPLEP (33)

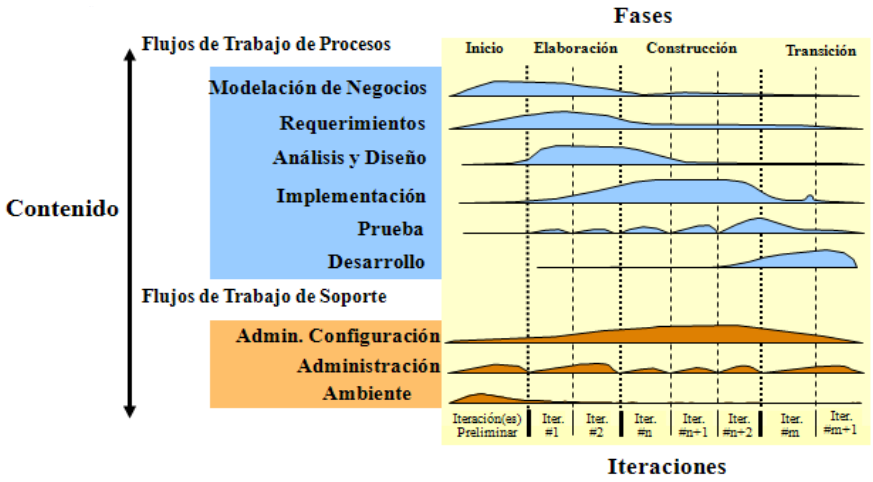


Anexo 8 Comparación entre los modelos LPS estudiados

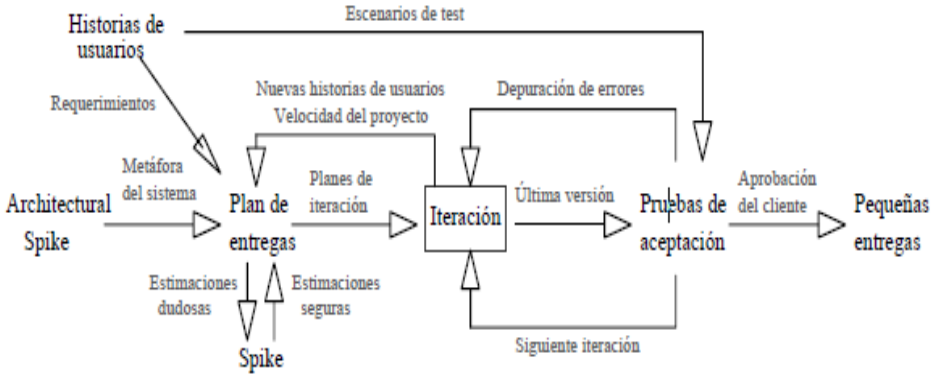
Indicadores / Modelos	Facilidad de asimilación y uso	Estándares	Repositorio	Arquitectura	Expediente	Valor
Modelo PuLSE	4	4	0	3	3	14
Metodología KobrA	2	2	2	4	2	12
Proceso de desarrollo PRAISE	4	3	0	4	2	13
Modelo de referencia CAFÉ	4	3	0	3	4	14
Modelo TWIN	3	2	4	3	2	14
Modelo WATCH	3	4	2	2	3	14

Modelo SPLEP	2	4	4	1	1	12
Modelo propuesto por el SEI	3	4	3	3	3	16
Modelo LPS desarrollo GESPRO	3	2	4	4	2	15

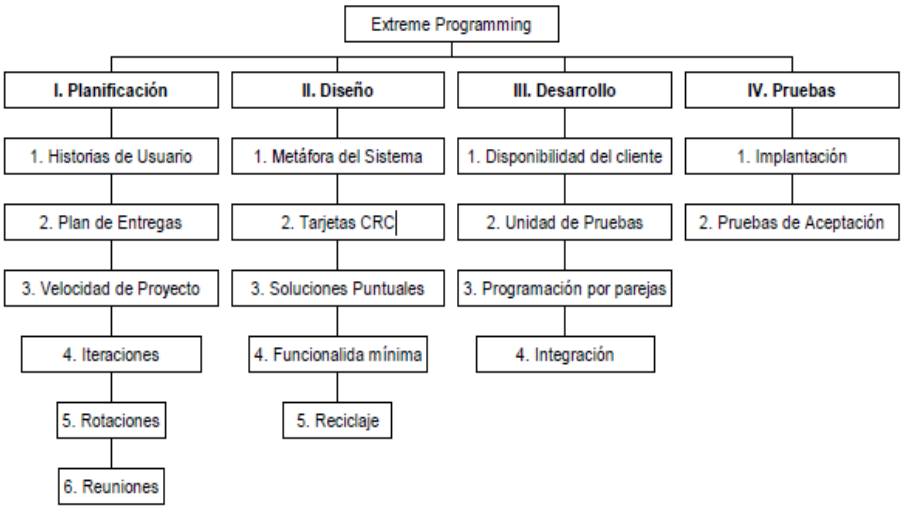
Anexo 9 Fases y flujos de trabajo en RUP (42)



Anexo 10 Ciclo de vida ideal de XP (45)

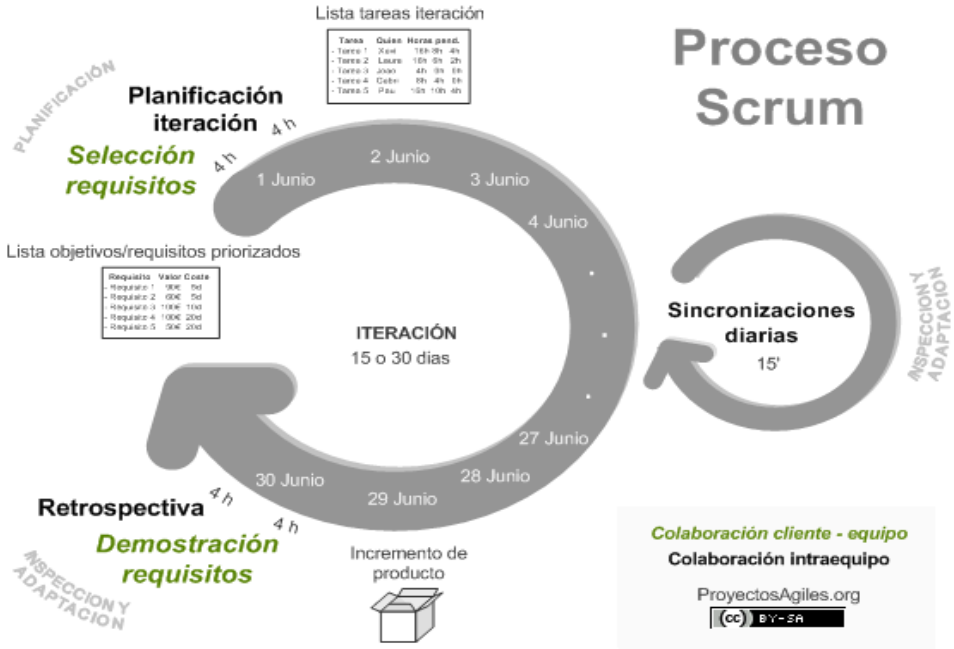


Anexo 11 Fases de la metodología XP (45)



Anexo 12 Fases del ciclo de vida de la metodología MSF (49)

Anexo 13Proceso de desarrollo SCRUM (54)



Anexo 14 Comparación entre metodología tradicional y metodología ágil. (63) (43) (49)

Criterios de Comparación	Desarrollo Tradicional	Desarrollo Ágil
Equipos	Especializado	Multidisciplinario
Fases	Fases Independientes	Fases Solapadas
Contrato	Existe un contrato prefijado	No existe un contrato tradicional, debe ser bastante flexible
Alcance	Seguimiento del Plan	Adaptación a los Cambios
Requisitos	Detallados	Visión del Producto
Artefactos	Más Artefactos. El modelado es esencial, mantenimiento de modelos.	Pocos Artefactos. El modelado es prescindible, modelos desechables.
Roles	Más Roles específicos.	Pocos Roles, más genéricos y flexibles.
Cliente	El cliente interactúa con el equipo de desarrollo mediante reuniones.	Cliente es parte del equipo de desarrollo.
Proyectos/equipos	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos.	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio.
Arquitectura	Se promueve que la arquitectura se defina tempranamente en el proyecto.	La arquitectura se va definiendo y mejorando a lo largo del proyecto.
Aspectos significativos	Énfasis en la definición del proceso: roles, actividades y artefactos.	Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.
Cambios	Se espera que no ocurran cambios de gran impacto durante el proyecto.	Se esperan cambios durante el proyecto

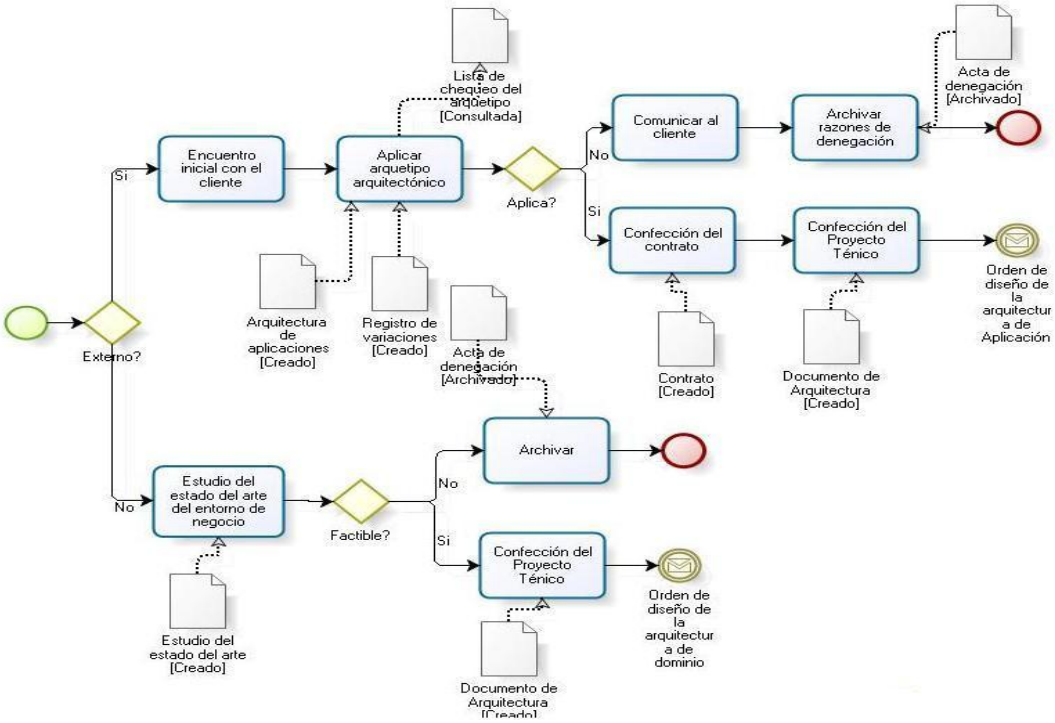
Anexo 15 Comparación entre metodologías ágiles.

CRITERIOS	CRYSTAL	DSDM	SCRUM	XP

Explicación	Activa con respecto a Orange y Clear, en construcción en cuanto a familia de metodologías incompletas.	Siendo una de las metodologías con mayor adopción en el Reino Unido y disponiendo de un gran número de experiencias se considera una metodología activa.	Una de las metodologías más antiguas y más utilizadas últimamente, actualmente podemos encontrar muchos estudios y experiencias con SCRUM.	Desde 1999 se ha convertido en la punta del iceberg de las metodologías ágiles, es la primera en internet, en adopciones y experiencias y la primera en estudios.
Estado	En construcción/Activa	Activa	Activa	Activa
Guías prescriptivas vs guías ajustables.	Prescriptivas	Ajustables	Ajustables	Ajustables
Calidad.	-Conformidad a los requisitos y consistencia y precisión (Fiabilidad), mediante pruebas de funcionalidad automática y regresiva. -Satisfacción del cliente, practica dos usuarios revisan y valoran las versiones liberadas. -Usabilidad, cumpliendo con el parámetro de claridad	-Conformidad a los requisitos, los documentos de especificación en DSDM deben tener criterios de calidad especificados, mediante las diferentes revisiones que se comprueban. -La fiabilidad del producto mediante su atributo de	-Conformidad a los requisitos. Mediante la utilización del rol: Product Owner y demos del producto. -Satisfacción del cliente, a través de las reuniones y demos presentadas.	-Satisfacción del cliente. Los clientes son una pieza clave y son requeridos en diferentes fases de XP. -Conformidad a los requisitos. En el juego de planificación, los clientes escogen objetivos.

	y precisión, existe una amplia documentación, ya que exigen generación de guía de usuario.	consistencia y precisión, con la realización de pruebas dinámicas y prototipos.		
Tamaño de equipos de trabajos	4 a 8	2 a 6	5 a 9	3 a 16

Anexo 16 Modelo de procesos fase de Concepción



Anexo 17 Especificación Proyecto Técnico

1. Proyecto técnico
1.1. Marco de referencia del proyecto
1.1.1.Antecedentes
1.1.2.Problemas a resolver
1.1.3.Solución del problema
1.1.4.Beneficiarios
1.1.5.Impactos esperados

1.2. Entorno de la Solución
1.2.1.Estructura y Organización
1.2.2.Marco legal
1.2.3.Entorno de explotación y uso de la solución
1.3. Formulación de la propuesta
1.3.1.Tipo de solución tecnológica
1.3.2.Objetivo General
1.3.3.Objetivos Específicos
1.3.4.Alcance
1.3.5.Criterios de éxito
1.3.6.Riesgos
1.4. Organización del Proyecto
1.4.1.Estructura del proyecto
1.4.2.Involucrados
1.4.3.Administración de Datos
1.5. Proyecto Solución de Software
1.5.1.Definición de la arquitectura del sistema
1.5.2.Arquitectura Información
1.5.3.Arquitectura tecnológica
1.5.4.Arquitectura de Sistema (Definiciones Macro)
1.5.5.Arquitectura de Despliegue (Definiciones Macro)
1.5.6.Arquitectura de Seguridad
1.5.7.Arquitectura de Integración e Interoperabilidad
1.5.8.Arquitectura de Datos y Flujos de Información
1.5.9.Alcance funcional de la solución de software
1.5.10. Atributos de Calidad Objetivos
1.5.11. Modelo de Desarrollo
1.5.12. Control de la Calidad
1.6. Despliegue
1.6.1.Objetivo del despliegue
1.6.2.Planificación del despliegue

1.6.3.Responsabilidades de las partes
1.6.4.Configuración y despliegue del equipamiento
1.7. Capacitación y Transferencia de conocimientos
1.7.1.Objetivos de la capacitación
1.7.2.Temáticas a abordar
1.7.3.Responsabilidades de las partes
1.8. Soporte Técnico de Software
1.9. Transferencia de tecnología
1.10. Productos entregables
1.11. Proyecto Solución de Equipamiento
1.11.1. Problemática a resolver
1.11.2. Objetivos Generales
1.11.3. Objetivos específicos
1.11.4. Alcance de la solución
1.11.5. Responsabilidades de las Partes
1.11.6. Garantía y Soporte Técnico
1.11.7. Entregables

Anexo 18 Especificación del Contrato

1. Contrato
1.1. Introducción
1.2. Comparecientes
1.3. Declaraciones
1.3.1.Declaración del proveedor
1.3.2.Declaración del cliente
1.3.3.Celebran las partes
1.4. Cláusulas
1.4.1.PRIMERA Definiciones
1.4.2.SEGUNDA Objeto del contrato
1.4.3.TERCERA Confidencialidad
1.4.4.CUARTA Causas eximentes de la responsabilidad
1.4.5.QUINTA Vigencia

1.4.6.SEXTA Modificación del acuerdo
1.4.7.SEPTIMA Valor Total del contrato y forma de pago
1.4.8.OCTAVA Terminación del contrato
1.4.9.NOVENA Garantías
1.4.10. DÉCIMA Lugar de prestación de los servicios
1.4.11. UNDÉCIMA Domicilio
1.4.12. DUODÉCIMA Disposiciones finales

Anexo 19 Especificación Estudio de Factibilidad.

1. Estudio de factibilidad
1.1. Áreas de procesos
1.2. Áreas funcionales
1.3. Regulaciones legislativas jurídicas
1.4. Normas, contratos y estándar
1.5. Agentes externos a la organización y elementos de información que intercambian
1.6. Niveles de informatización y sistemas de software de referencia

Anexo 20 Arquetipo Arquitectónico

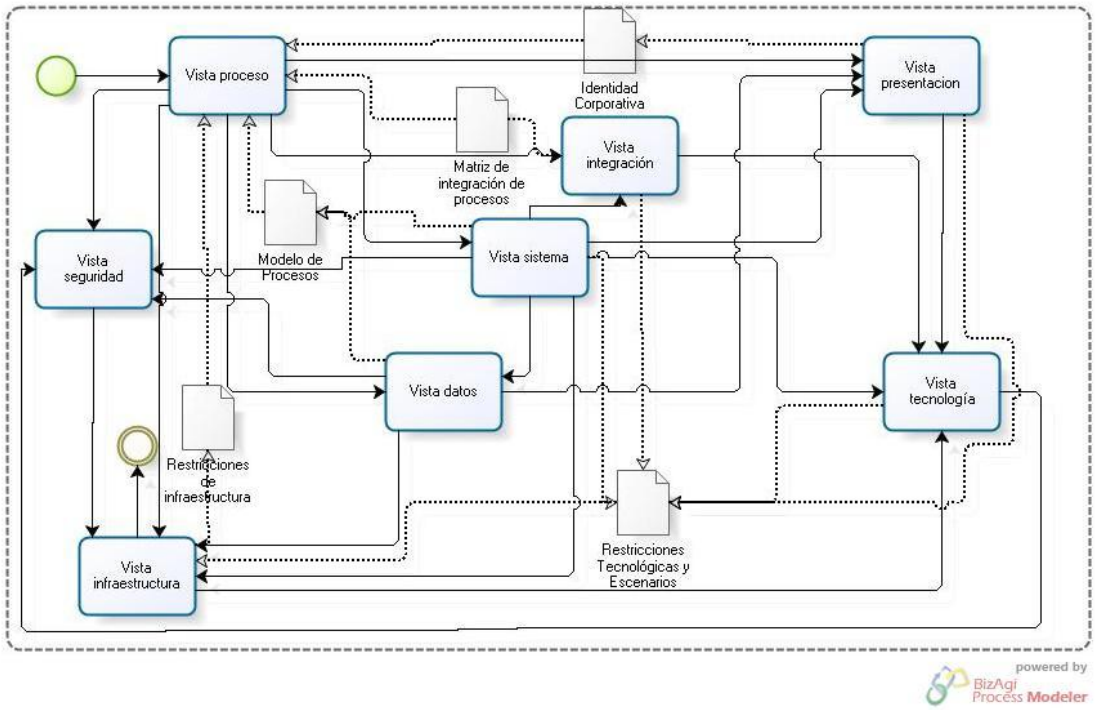
Plano de proceso
1. *Modelo de procesos de la arquitectura
1.1. Registro de macro procesos
1.1.1.Nombre del macro proceso
1.1.2.Modelo general
1.1.3.Clasificación (Clave, Soporte, Estratégico)
1.2. Registro procesos
1.2.1.Código del proceso
1.2.2.Descripción
1.2.3.Modelo especifico del proceso
1.2.4.Conceptos asociados
1.2.5.Reglas de negocio
1.2.6.Actividades automatizables
1.2.7.Servicios de entrada interno al macro proceso
1.2.7.1. Parámetros de entrada
1.2.7.2. Parámetros de salida
1.2.8.Servicios provisto
1.2.8.1. Parámetros de entrada
1.2.8.2. Parámetros de salida

1.2.9.Estándar funcional
1.3. Servicios de entrada externo al macro proceso
1.3.1.Nombre del servicio
1.3.1.1. Parámetros de entrada
1.3.1.2. Parámetros de salida
1.3.2.Servicios provisto
1.3.2.1. Nombre del servicio
1.3.2.2. Parámetros de entrada
1.3.2.3. Parámetros de salida
2. *Estructura jerárquica organizacional
3. *Mapa conceptual
1.1. Área de proceso
1.1.1.Proceso concreto
1.1.1.1. Registro de conceptos
1.1.1.1.1. Nombre del concepto
1.1.1.1.2. Descripción
1.1.1.1.3. Tipos de datos
1.1.1.1.3.1. Nombre del tipo
1.1.1.1.3.2. Tipo de dato
1.1.1.1.3.3. Requerido?
1.1.1.1.3.4. Clases de equivalencia
1.1.1.1.3.5. Clases invalidas
1.1.1.1.3.6. Asociado a
1.1.1.1.3.7. Nulo?
1.1.1.1.3.8. Valor por defecto
4. Estándares
1.2. Estándares de datos (Atributos, tipos de datos, estructura, precedencia, norma etc)
1.3. Estándares de información. (Informes, Proformas Reportes).
1.4. Estándar de interoperabilidad. (Soporte, formato, medio tecnológico, norma etc)
5. Reglas de negocio (Agrupada por área de proceso y proceso concreto)
Plano de sistema
1. Documento de especificación de la arquitectura de sistema
1.1. Modelo general (Esquema modular de las estructuras arquitectónicas del producto)
1.2. Registro de Subsistema
1.2.1.Diagrama de componentes del subsistema
1.2.2.Código Área de proceso que abstrae
1.2.3.Registro de componentes
1.2.3.1. Nombre del componente
1.2.3.2. Responsabilidad arquitectónica

1.2.3.3. Lista de procesos asociados (códigos separados por coma)
1.2.3.4. Lista de escenarios arquitectónicos que implementa (lista de códigos)
1.2.3.5. Diagrama de clase
1.2.3.6. Patrones arquitectónicos aplicable
1.2.3.7. Estándar funcional que implementa (Código del estándar)
1.2.3.8. Estándar de interoperabilidad que implementa (Código del estándar)
1.2.3.9. Registro de Enumerados
1.2.3.10. Registro de constantes
1.2.3.11. Variables globales que demanda
1.2.4.Servicios provisto
1.2.4.1. Nombre o Código del servicio
1.2.4.1.1. Parámetros de entrada
1.2.4.1.2. Parámetros de salida
1.2.4.1.3. Código o Referencia del estándar que implementa
1.2.5.Servicios requeridos del exterior
1.2.5.1. Nombre o Código del servicio
1.2.5.1.1. Parámetros de entrada
1.2.5.1.2. Parámetros de salida
1.2.5.1.3. Código o Referencia del estándar que implementa
2. *Especificación de requisitos funcionales de la arquitectura de sistema
3. Agrupación funcional
3.1. Nombre
3.2. Condigo del proceso asociado
3.3. Registro de escenarios arquitectónicos funcionales
3.3.1.Código requisito
3.3.2.Descripción
3.3.3.Dependencias
3.3.4.Pre condiciones
3.3.5.Descripción
3.3.6.Ref a reglas de negocio asociadas
3.3.7.Atributos de calidad esperados
3.3.7.1. Nombre atributo
3.3.7.2. Estimulo
3.3.7.3. Indicadores esperados
3.3.7.3.1. Indicador
3.3.7.3.2. Valor esperado
3.3.7.3.3. Valores impropios
3.3.8.Post condiciones
3.3.9.Lista de restricciones arquitectónicas del escenario (Lista de cód. separados por coma)

Plano de Información
1. Identidad de la organización
2. Papelería , proformas y estilos
3. Normas y estándares de representación de la información
4. Arquitectura documental
Plano de infraestructura
1. Infraestructura tecnológica (redes, capacidades de almacenamiento, etc)
2. Servicios telemáticos
3. Infraestructura física
4. Distribución geográfica

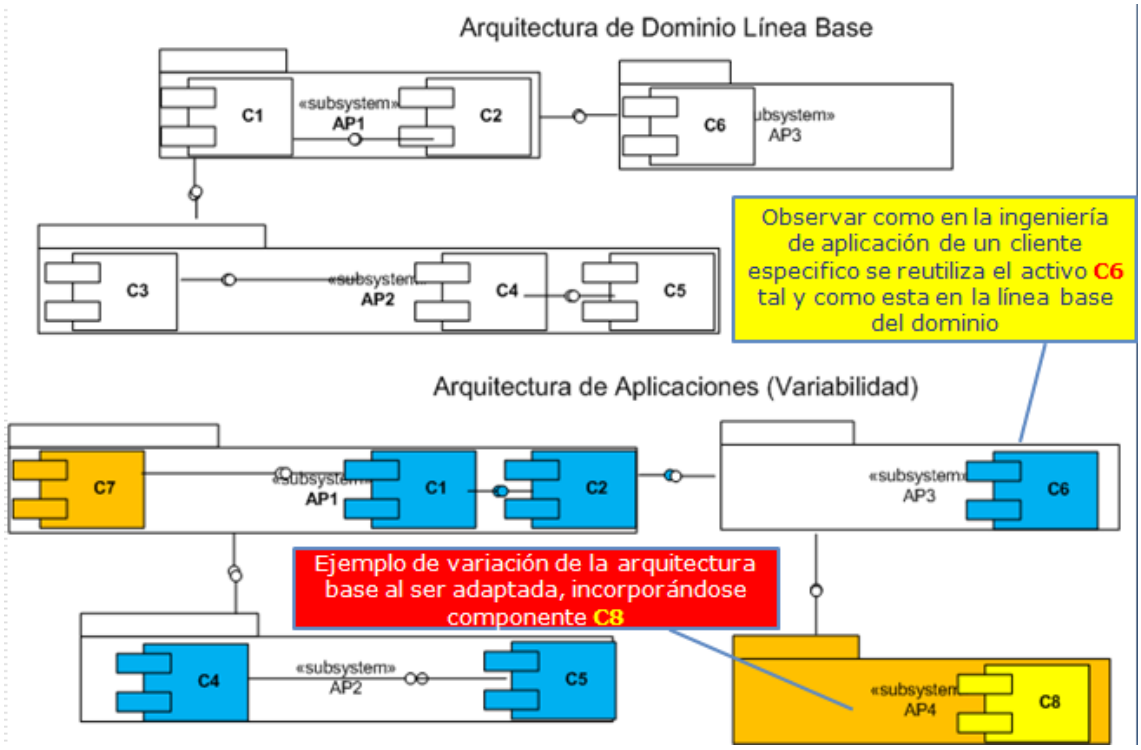
Anexo 21 Relación entre las vistas del Modelo arquitectónico UCI



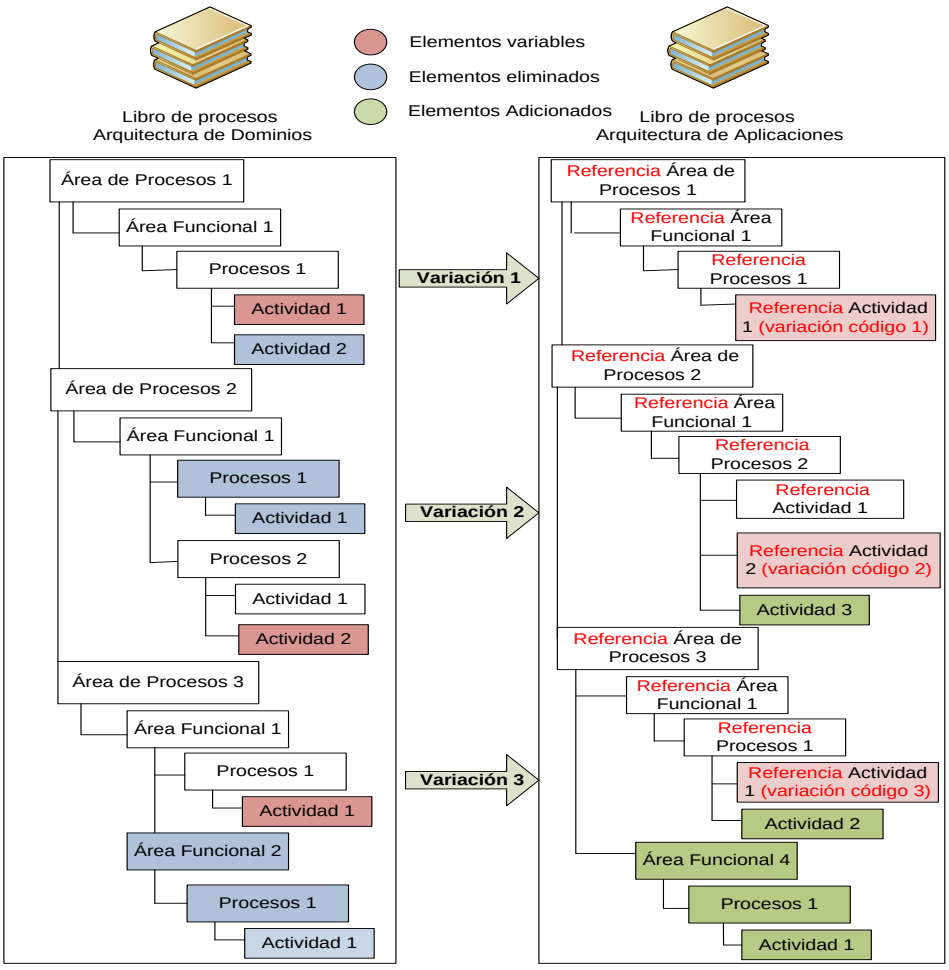
La vista de Procesos tributa elementos para identificar escenarios y restricción a la vista de sistema, integración y datos fundamentalmente, aunque aporta entradas analíticas al resto de las vistas. La vista de Sistema provee información y consideraciones analíticas a la vista de integración, la vista de tecnología, la vista de datos y la vista de infraestructura. Por otra parte la vista de Integración demanda hitos de solución a la vista Tecnológica. Esta a su vez parte de los

elementos analíticos provistos por la vista de sistema, de integración y seguridad. La vista de Presentación usa los elementos analíticos provistos por la vista de procesos, sistema y datos. También demanda hitos de solución arquitectónica a la vista de tecnología. En el caso de la vista de Seguridad parte de los elementos analíticos provistos por la vista de procesos, sistema y datos. La vista de seguridad también demanda hitos de solución arquitectónica a la vista de tecnología. Por último la vista de Despliegue o infraestructura toma como fuente originaria los elementos analíticos provistos por la vista de procesos, sistema y datos. A la vez demanda hitos de solución arquitectónica a la vista de tecnología.

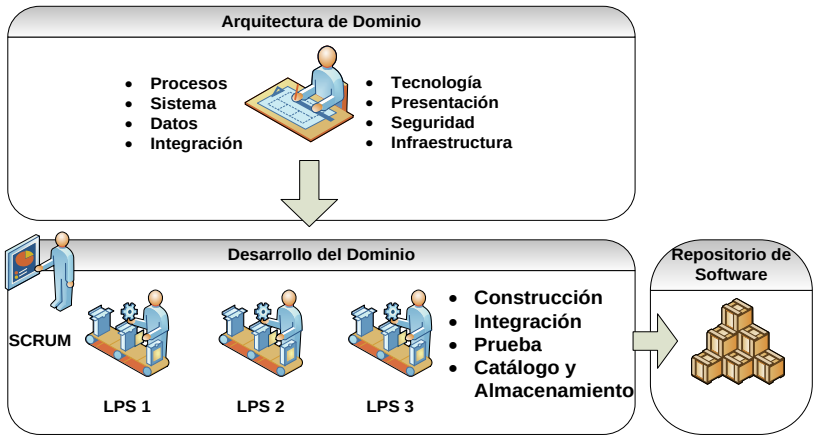
Anexo 22 Ejemplo instanciación de la Arquitectura de Dominio por la Arquitectura de Aplicaciones



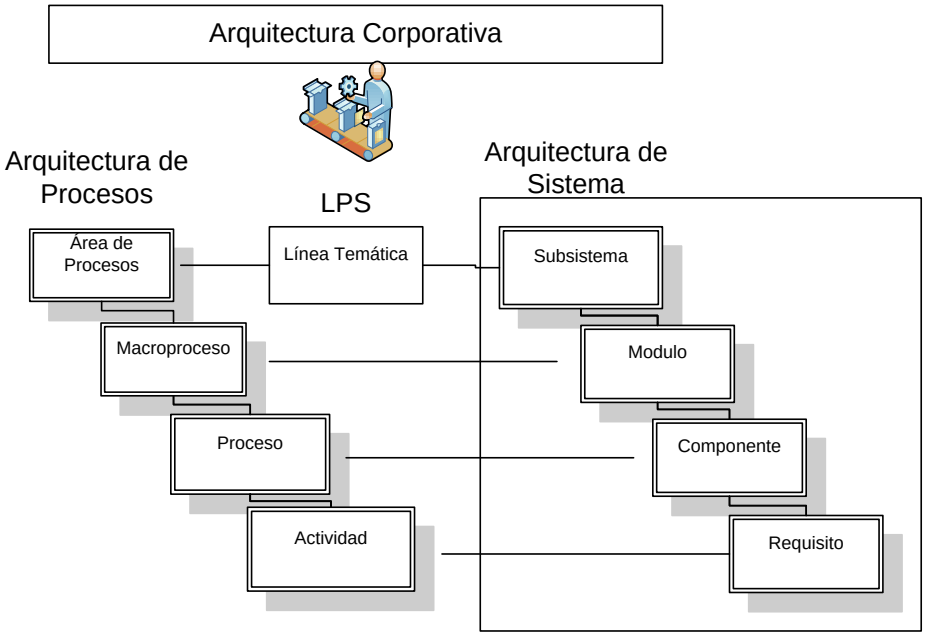
Anexo 23 Ejemplo de definición de la Arquitectura de Aplicaciones. Plano de procesos



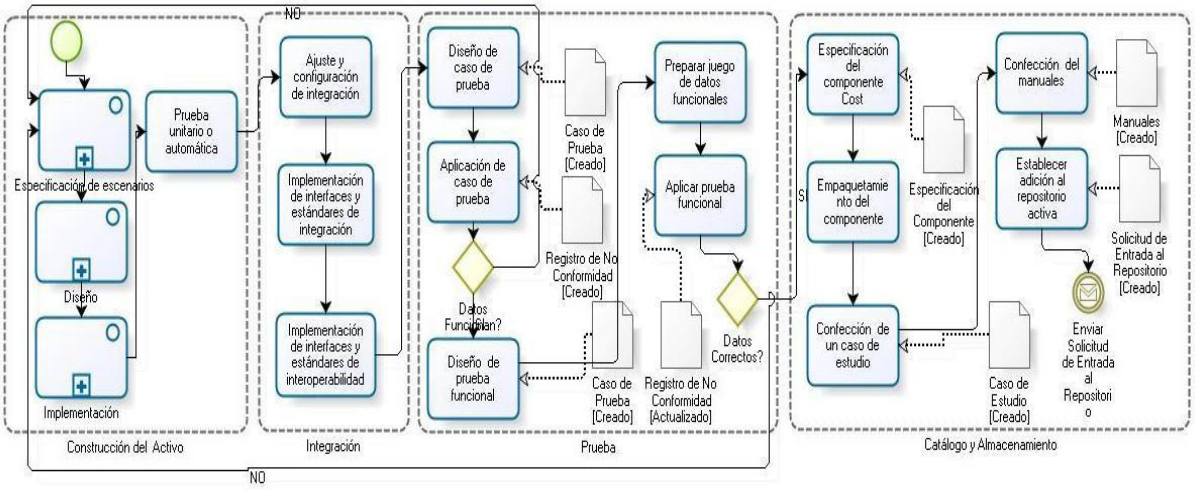
Anexo 24 Representación de la comunicación de la Arquitectura de Dominio con la Construcción del Dominio



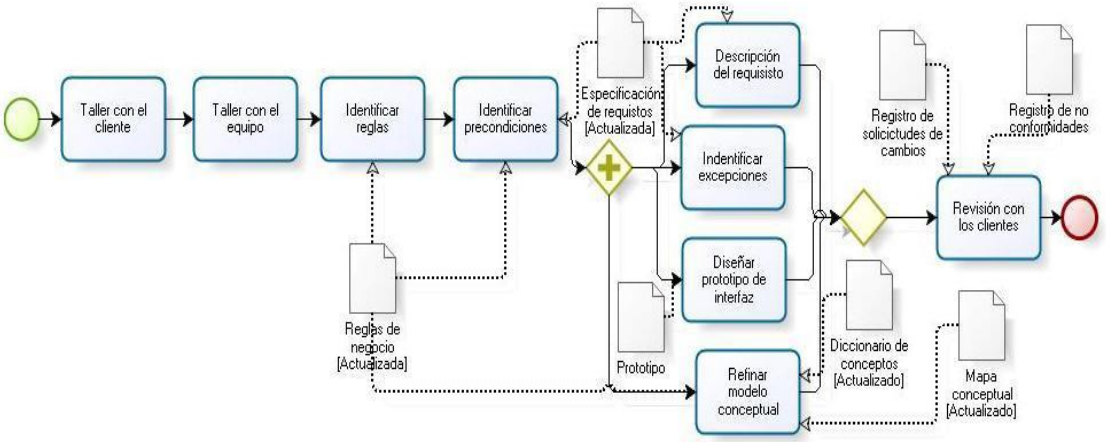
Anexo 25 Matriz de proyección de equivalencia de la Vista de Procesos a la Vista de Sistema.



Anexo 26 Proceso de Construcción de Dominio



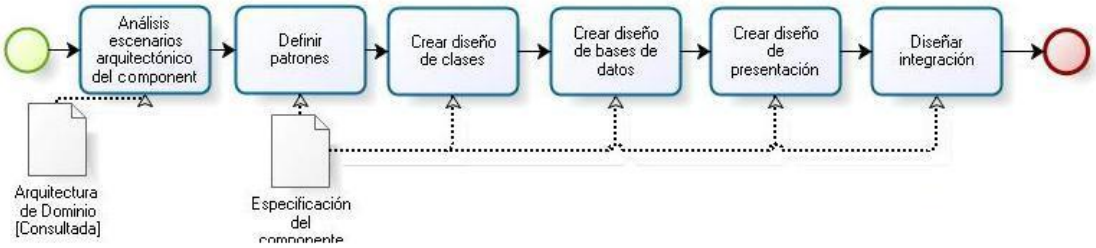
Anexo 27 Proceso de Especificación de escenarios



Anexo 28 Tareas y artefactos del proceso Especificación de escenarios

Tareas	**Artefactos**
Taller cliente	
Taller equipo	
Identificar Reglas	Reglas de negocio
Descripción del requisito	Especificación de requisitos
Refinar modelo conceptual	Reglas de negocio, Diccionario de conceptos, Mapa conceptual
Diseñar prototipo interfaz	Prototipo
Identificar excepciones	Especificación de requisitos
Identificar precondiciones	Reglas de negocio, Especificación de requisitos
Revisión con los clientes	Registro de solicitudes de cambios, Registro de no conformidades

Anexo 29 Proceso de Diseño



Anexo 30 Proceso de implementación

Anexo 31 Especificación del componente

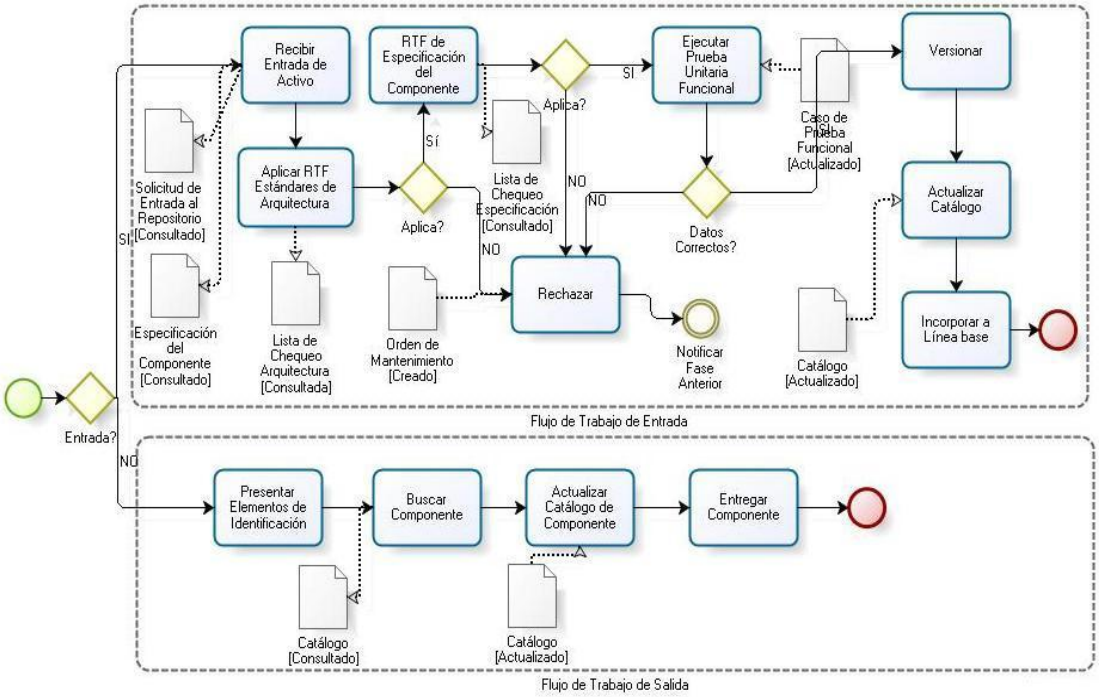
1. Especificación del componente
1.1. Código
1.2. Nombre
1.3. Descripción
1.4. Revisión
1.5. Detalles de la revisión
1.6. Responsabilidad funcional
1.6.1.Áreas de procesos relacionadas
1.6.2.Procesos relacionados
1.6.2.1. Requisitos que implementa
1.7. Interfaces de interoperabilidad
1.8. Estándares de interoperabilidad
1.9. Interfaces de integración
1.10. Patrones de implementación
1.11. Modelo de clases e interfaces
1.12. Controles de presentación
1.12.1. Nombre
1.12.2. Descripción
1.12.3. Prototipo
1.12.4. Interfaces
1.13. Diseño de persistencia
1.13.1. Modelo
1.13.2. Scrip
1.13.3. Datos de carga inicial
1.13.4. Matriz de dependencia de datos
1.14. Matriz de dependencia funcional
1.14.1. Componentes tecnológicos
1.14.2. Infraestructura
1.15. Casos de estudio
1.16. Manuales

1.16.1. Manual de instalación
1.16.2. Manual de configuración
1.16.3. Manual de usuario
1.16.4. Manual de desarrollador
2. Costo
3. Registro de reutilización
3.1.1.Aplicación
3.1.2.Fecha
3.1.3.Versión
3.1.4.Revisión
3.1.5.Valor comercial

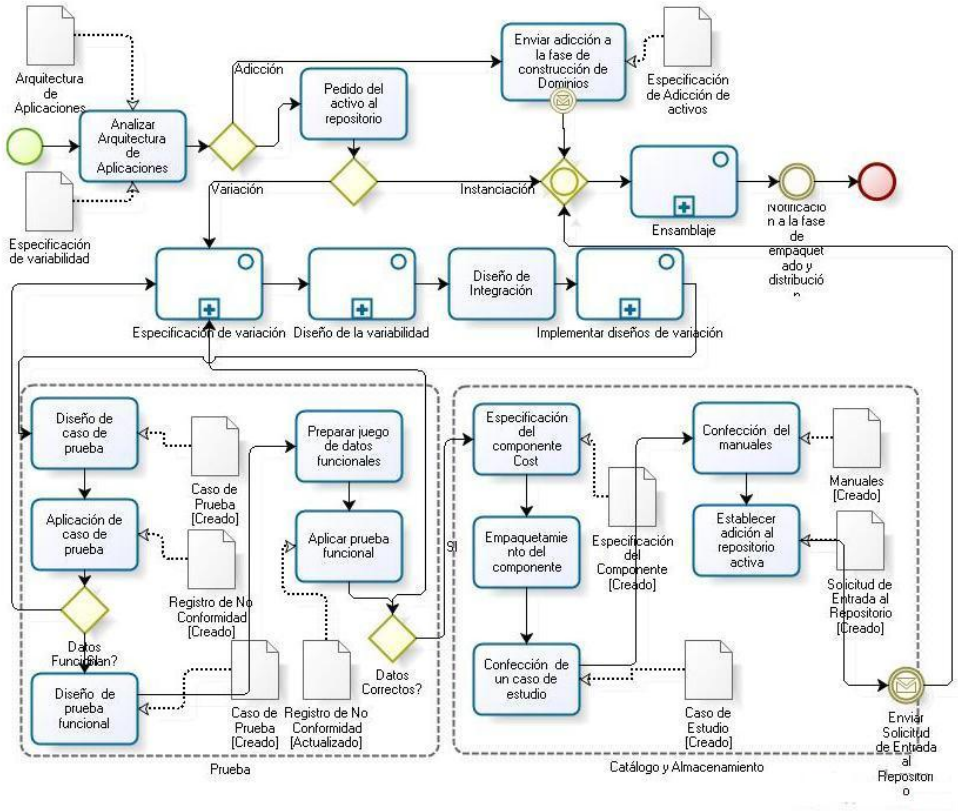
Anexo 32 Registro de no Conformidad

1. Registro de no Conformidad
1.1. Código
1.2. Fecha recepción
1.3. Fecha cierre
1.4. Estado
1.5. Emitida por
1.6. Descripción
1.7. Propuesta de solución
2. Registro de Solicitud de cambio
2.1. Código
2.2. Fecha recepción
2.3. Fecha cierre
2.4. Descripción
2.5. Estado
2.6. A cargo de
2.7. Propuesta de solución
2.8. Referencia al sistema o parte de este
2.9. Prioridad
2.10. Costo

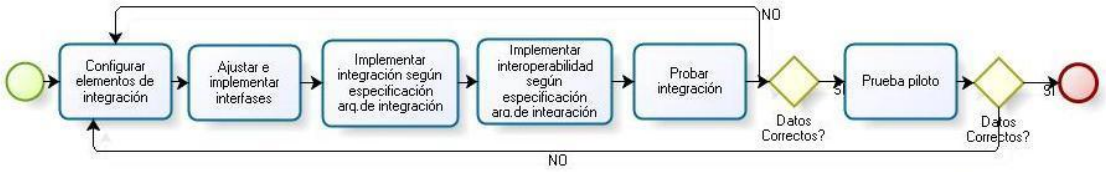
Anexo 33 Procesos de Entrada y Salida de Repositorio de Activos de Software



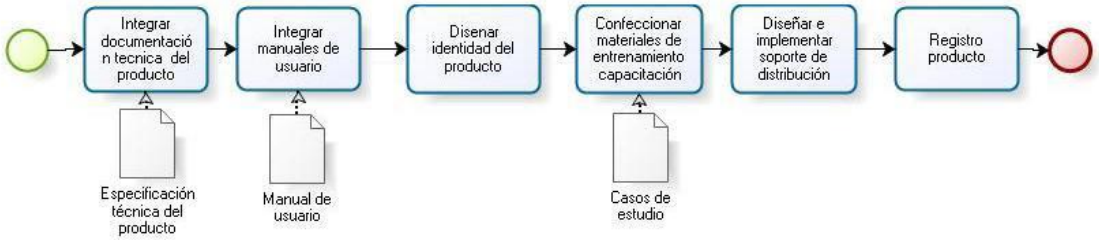
Anexo 34 Proceso Construcción de Aplicaciones



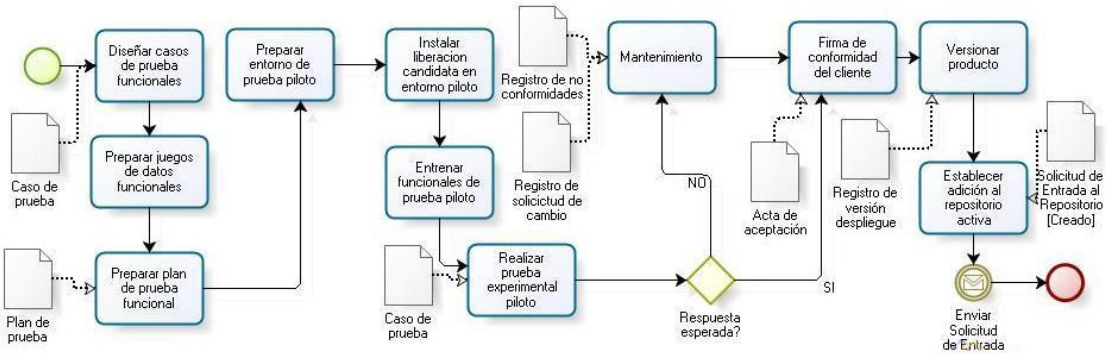
Anexo 35 Proceso de Ensamblaje



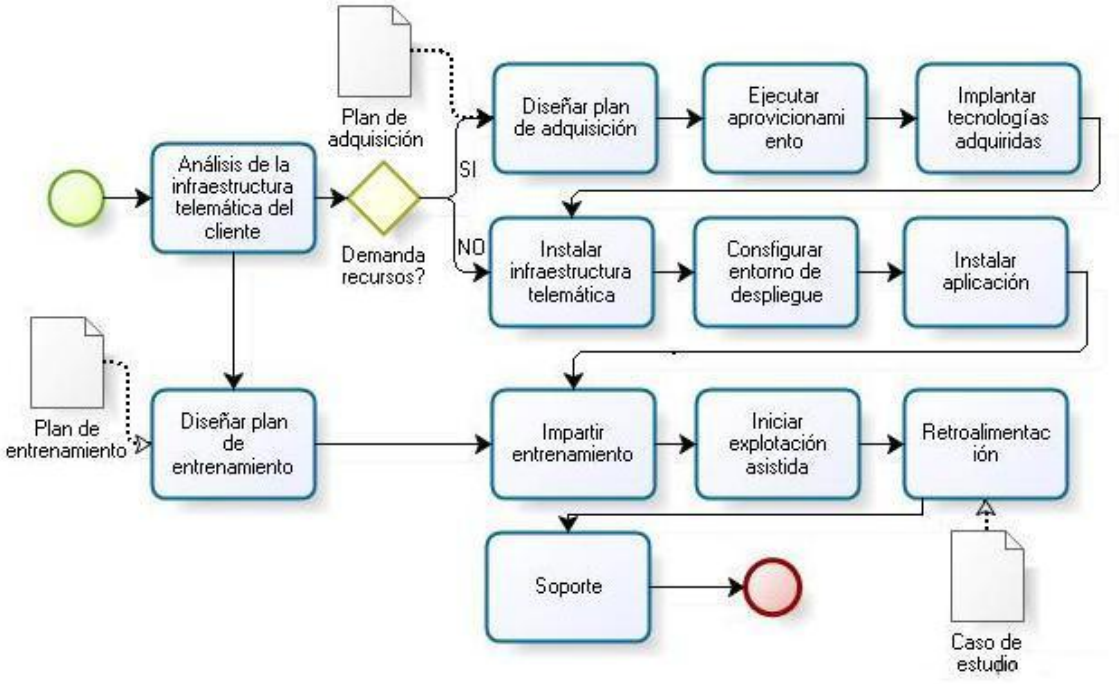
Anexo 36 Proceso de Empaquetado



Anexo 37 Proceso de Liberación



Anexo 38 Proceso de Despliegue

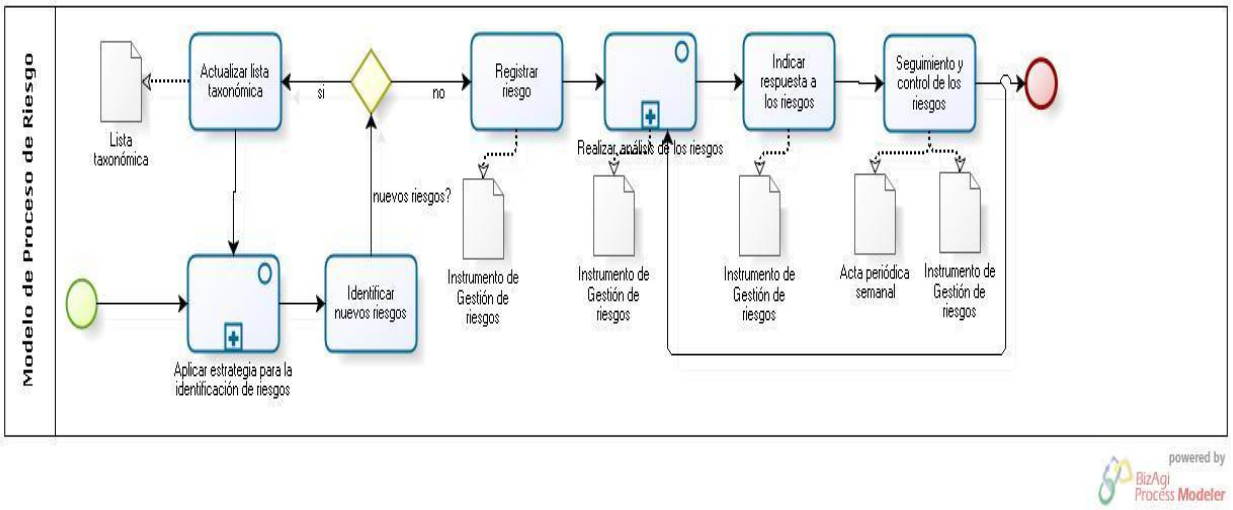


Anexo 39 Rores y responsabilidades del Método de desarrollo de la LPS basado en SRUM

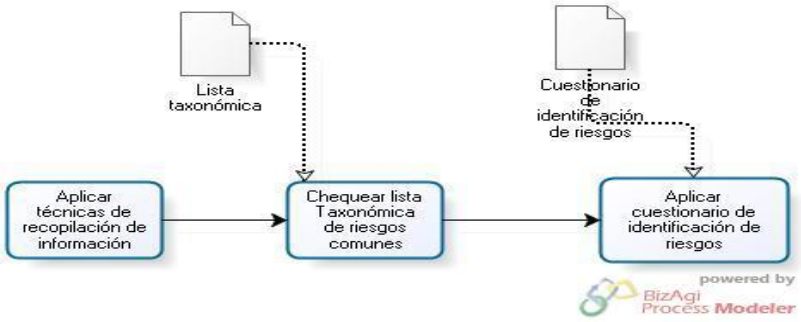
Roles	Responsabilidades
Jefe de LPS	Responsable de garantizar los cronogramas y compromisos de la LPS Supervisar el proceso de desarrollo. Identificar y gestionar nuevos negocios.
Jefe de unidad de negocio (SCRUM Master)	Es el responsable legal de cara al cliente de todos los proyectos de la UN. Gestiona los documentos legales de los proyectos de la UN. Responsable de organizar y controlar el trabajo de la UN según las fases que propone SCRUM para el desarrollo de SW y las herramientas existentes en el área para la gestión de proyectos. Mantener actualizado el cronograma en la herramienta de gestión de proyectos y garantizar su ejecución. Planificar y controlar las tareas de los miembros de la UN. Realizar los planes de trabajo Individuales. Evaluar a los miembros de la UN mensualmente. Llevar las actas de las reuniones y talleres. Identificar, describir y validar los procesos de negocio y los requisitos de software asociados a estos. Elaborar el Mapa de Procesos de la UN utilizando BPMN. Elaborar la Descripción de Procesos de Negocio. Elaborar arquetipos de la UN. Planificar talleres de diseño. Coordinar el proceso de diseño de casos de prueba. Coordinar las pruebas de aceptación o liberación. Revisar, controlar la aplicación de las normas y estándares de calidad que se establecen en la empresa. Gestiona los componentes de su UN en el repositorio.
Arquitecto de Sistema	Responsable de que se cumplan las políticas y estándares definidos en la Arquitectura. Define los métodos de integración en el proyecto y la Arquitectura del Sistema. Modera el Taller de Diseño. Gestiona los componentes comunes a varias UN en el repositorio.
Desarrollador (SCRUM Developer)	Construye y actualiza el Modelo de dominio, además responde por el manejo y recuperación de la información del mismo. Elabora la documentación técnica del proyecto Construye los componentes de software de la UN. Ensambla soluciones utilizando los componentes existentes en el

	repositorio. Diseña casos de prueba. Realiza pruebas a los componentes.

Anexo 40 Modelo de Procesos de Gestión de Riesgos



Anexo 41 Estrategia para la Identificación de Riesgos



Anexo 42 Lista Taxonómica de Riesgos

A. Elaboración de la Planificación
A.1. Las definiciones de la planificación, de los recursos y del producto han sido impuestas por el cliente o un directivo superior, y no están equilibradas.
A.2. Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).
A.3. La planificación no incluye tareas necesarias.
A.4. La planificación se ha basado en la utilización de personas específicas de un equipo, pero estas personas no están disponibles.
A.5. No se puede construir un producto de tal envergadura en el tiempo asignado.
A.6. El producto es más grande que el estimado (en líneas de código, en el número de puntos

función, o en relación con el tamaño del proyecto anterior).
A.7. El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).
A.8. La reestimación debida a un retraso en la planificación es demasiado optimista o ignora la historia del proyecto.
A.9. La presión excesiva en la planificación reduce la productividad.
A.10. La fecha final ha cambiado sin ajustarse al ámbito del producto o a los recursos disponibles.
A.11. Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.
A.12. Las áreas desconocidas del producto llevan más tiempo del esperado en el diseño y en la implementación.
B. Organización y Gestión
B.1. El proyecto carece de un promotor efectivo en los superiores.
B.2. El proyecto languidece demasiado en el inicio difuso.
B.3. Los despidos y las reducciones de la plantilla reducen la capacidad del equipo.
B.4. Dirección o marketing insisten en tomar decisiones técnicas que alargan la planificación. B.5. La estructura inadecuada de un equipo reduce la productividad.
B.6. El ciclo de revisión/decisión de la directiva es más lento de lo esperado.
B.7. El presupuesto varía el plan del proyecto.
B.8. La dirección toma decisiones que reducen la motivación del equipo de desarrollo.
B.9. Las tareas no técnicas encargadas a terceros necesitan más tiempo del esperado (aprobación del presupuesto, aprobación de la adquisición de material, revisiones legales, seguridad, etc.).
B.10. La planificación es demasiado mala para ajustarse a la velocidad de desarrollo deseada.
B.11. Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente.
B.12. La dirección pone más énfasis en las heroicidades que en informarse exactamente del estado, lo que reduce su habilidad para detectar y corregir problemas.
C. Ambiente/Infraestructura de Desarrollo
C.1. Los espacios no están disponibles en el momento necesario.
C.2. Los espacios están disponibles pero no son adecuados (por ejemplo, falta de teléfonos, cableado de la red, mobiliario, material de oficina, etc.).
C.3. Los espacios están sobre-utilizados, son ruidosos o distraen.

C.4. Las herramientas de desarrollo no están disponibles en el momento deseado.
C.5. Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.
C.6. Las herramientas de desarrollo no se han elegido en función de sus características técnicas, y no proporcionan las prestaciones previstas.
C.7. La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.
D. Usuarios Finales
D.1. Los usuarios finales insisten en nuevos requisitos.
D.2. En el último momento, a los usuarios finales no les gusta el producto, por lo que hay que volver a diseñarlo y a construirlo.
D.3. Los usuarios no han realizado la compra del material necesario para el proyecto y, por tanto, no tienen la infraestructura necesaria.
D.4. No se ha solicitado información al usuario, por lo que el producto al final no se ajusta a las necesidades del usuario, y hay que volver a crear el producto.
E. Cliente
E.1. El cliente insiste en nuevos requisitos.
E.2. Los ciclos de revisión/decisión del cliente para los planes, prototipos y especificaciones son más lentos de lo esperado.
E.3. El cliente no participa en los ciclos de revisión de los planes, prototipos y especificaciones, o es incapaz de hacerlo, resultando unos requisitos inestables y la necesidad de realizar unos cambios que consumen tiempo.
E.4. El tiempo de comunicación del cliente (por ejemplo, tiempo para responder a las preguntas para aclarar los requisitos) es más lento del esperado.
E.5. El cliente insiste en las decisiones técnicas' que alargan la planificación.
E.6. El cliente intenta controlar el proceso de desarrollo, con lo que el progreso es más lento de lo esperado.
E.7. Los componentes suministrados por el cliente no son adecuados para el producto que se está desarrollando, por lo que se tiene que hacer un trabajo extra de diseño e integración.
E.8. Los componentes suministrados por el cliente tienen poca calidad, por lo que tienen que hacerse trabajos extra de comprobación, diseño e integración.
E.9. Las herramientas de soporte y entornos impuestos por el cliente son incompatibles, tienen un

bajo rendimiento o no funcionan de forma adecuada, con lo que se reduce la productividad.
E.10. El cliente no acepta el software entregado, incluso aunque cumpla todas sus especificaciones.
E.11. El cliente piensa en una velocidad de desarrollo que el personal de desarrollo no puede alcanzar.
F. Personal Contratado
F.1. El personal contratado no suministra los componentes en el período establecido.
F.2. El personal contratado proporciona material de una calidad inaceptable, por lo que hay que añadir un tiempo extra para mejorar la calidad.
F.3. Los proveedores no se integran en el proyecto, con lo que no se alcanza el nivel de rendimiento que se necesita.
G. Requisitos
G.1. Los requisitos se han adaptado, pero continúan cambiando.
G.2. Los requisitos no se han definido correctamente. y su redefinición aumenta el ámbito del proyecto.
G.3. Se añaden requisitos extra.
G.4. Las partes del proyecto que se no se han especificado claramente consumen más tiempo del esperado.
H. Producto
H.1. Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.
H.2. Una calidad no aceptable requiere de un trabajo de comprobación, diseño e implementación superior al esperado.
H.3. Utilizar lo último en informática alarga la planificación de forma impredecible.
H.4. El desarrollo de funciones software erróneas requiere volver a diseñarlas y a implementarlas.
H.5. El desarrollo de una interfaz de usuario inadecuada requiere volver a diseñarla y a implementarla.
H.6. El desarrollo de funciones software innecesarias alarga la planificación.
H.7. Alcanzar el ámbito del producto o las restricciones de velocidad requiere más tiempo del esperado, incluyendo el tiempo para volver a diseñar e implementar.
H.8. Unos requisitos rígidos de compatibilidad con el sistema existente necesitan un trabajo extra

de comprobación, diseño e implementación.
H.9. Los requisitos para crear interfaces con otros sistemas, otros sistemas complejos, u otros sistemas que no están bajo el control del equipo de desarrollo suponen un diseño, implementación y prueba no previstos.
H.10.El requisito de trabajar con varios sistemas operativos necesita más tiempo del esperado.
H.11.El trabajo con un entorno software desconocido causa problemas no previstos.
H.12.El trabajo con un entorno hardware desconocido causa problemas imprevistos.
H.13.El desarrollo de un tipo de componente nuevo para la organización consume más tiempo del esperado.
H.14.Depender de una tecnología que aún está en fase de desarrollo alarga la planificación.
I. Fuerzas mayores
I.1. El producto depende de las normativas del gobierno, que pueden cambiar de forma inesperada.
I.2. El producto depende de estándares técnicos provisionales, que pueden cambiar de forma inesperada.
J. Personal
J.1. La contratación tarda más de lo esperado.
J.2. Las tareas preliminares (por ejemplo, formación, finalización de otros proyectos, adquisición de licencias) no se han completado a tiempo.
J.3. La falta de relaciones entre la dirección y el equipo de desarrollo ralentiza la toma de decisiones.
J.4. Los miembros del equipo no se implican en el proyecto, y por lo tanto no alcanzan el nivel de rendimiento deseado.
J.5. La falta de motivación y de moral reduce la productividad.
J.6. La falta de la especialización necesaria aumenta los defectos y la necesidad de repetir el trabajo.
J.7. El personal necesita un tiempo extra para acostumbrarse a trabajar con herramientas o entornos nuevos.
J.8. El personal necesita un tiempo extra para acostumbrarse a trabajar con hardware nuevo.
J.9. El personal necesita un tiempo extra para aprender un lenguaje de programación nuevo.
J.10. El personal contratado abandona el proyecto antes de su finalización.

J.11. Alguien de la plantilla abandona el proyecto antes de su finalización.
J.12. La incorporación de nuevo personal de desarrollo al proyecto ya avanzado, y el aprendizaje y comunicaciones extra imprevistas reducen la eficiencia de los miembros del equipo existentes.
J.13. Los miembros del equipo no trabajan bien juntos.
J.14. Los conflictos entre los miembros del equipo conducen a problemas en la comunicación y en el diseño, errores en la interfaz y tener que repetir algunos trabajos.
J.15. Los miembros problemáticos de un equipo no son apartados, influyendo negativamente en la motivación del resto del equipo.
J.16. Las personas más apropiadas para trabajar en el proyecto no están disponibles.
J.17. Las personas más apropiadas para trabajar en el proyecto están disponibles, pero no se pueden incorporar por razones políticas o de otro tipo.
J.18. Se necesitan personas para el proyecto con habilidades muy específicas y no se encuentran.
J.19. Las personas clave sólo están disponibles una parte del tiempo.
J.20. No hay suficiente personal disponible para el proyecto.
J.21. Las tareas asignadas al personal no se ajustan a sus posibilidades.
J.22. El personal trabaja más lento de lo esperado.
J.23. El sabotaje por parte de la dirección del proyecto deriva en una planificación ineficiente e inefectiva.
J.24. El sabotaje por parte del personal técnico deriva en una pérdida de trabajo o en un trabajo de poca calidad, por lo que hay que repetir algunos trabajos.
K. Diseño e Implementación
K.1. Un diseño demasiado sencillo no cubre las cuestiones principales, con lo que hay que volver a diseñar e implementar.
K.2. Un diseño demasiado complejo exige tener en cuenta complicaciones innecesarias e improductivas en la implementación.
K.3. Un mal diseño implica volver a diseñar e implementar.
K.4. La utilización de metodologías desconocidas deriva en un periodo extra de formación y tener que volver atrás para corregir los errores iniciales cometidos en la metodología.
K.5. El producto está implementado en un lenguaje de bajo nivel (por ejemplo, ensamblador) y la productividad es menor de la esperada.
K.6. No se puede implementar la funcionalidad deseada con el lenguaje o bibliotecas utilizados: el

personal de desarrollo tiene que utilizar otras bibliotecas, o crearlas él mismo para conseguir la funcionalidad deseada.
K.7. Las bibliotecas de código o clases tienen poca calidad, y generan una comprobación extra, corrección de errores y la repetición de algunos trabajos.
K.8. Se ha sobreestimado el ahorro en la planificación derivado del uso de herramientas para mejorar la productividad.
K.9. Los componentes desarrollados por separado no se pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos.
L. Proceso
L.1. La burocracia produce un progreso más lento del esperado.
L.2. La falta de un seguimiento exacto del progreso hace que se desconozca que el proyecto esté retrasado hasta que está muy avanzado.
L.3. Las actividades iniciales de control de calidad son recortadas, haciendo que se tenga que repetir el trabajo.
L.4. Un control de calidad inadecuado hace que los problemas de calidad que afectan a la planificación se conozcan tarde.
L.5. La falta de rigor (ignorar los fundamentos y estándares del desarrollo de software) conduce a fallos de comunicación, problemas de calidad y repetición del trabajo. Un consumo de tiempo innecesario.
L.6. El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario.
L.7. La creación de informes de estado a nivel de directiva lleva más tiempo al desarrollador de lo esperado.
L.8. La falta de entusiasmo en la Gestión de Riesgos impide detectar los riesgos más importantes del proyecto.
L.9. La Gestión de Riesgos del proyecto software consume más tiempo del esperado.

Anexo 43 Cuestionarios para la Identificación de Riesgos

Cuestionarios para la Identificación de Riesgos
A. Ingeniería
1. Requerimientos
a. Estabilidad

¿Mientras se desarrolla el producto hay requerimientos cambiantes?
¿Son estables los requerimientos?
(No) (1.a) ¿Cual es el efecto que provoca en el sistema los requerimientos cambiantes?
Calidad
Funcionalidad
Planificación
Integración
Diseño
Pruebas
[2] ¿Están cambiando las interfaces externas?
b. Completamiento.
[¿Hay requerimientos faltantes o están incompletas las especificaciones?]
[3] ¿En las especificaciones quedan elementos por ser definidos?
[4] ¿Conoce algunos requerimientos que deben estar en las especificaciones y no están?
(Si) (4.a) ¿Es posible realizar estos requerimientos en el sistema?
[5] ¿El cliente tiene requerimientos o expectativas que no han sido documentadas?
(Si) (5.a) ¿Hay alguna forma de capturar esos requerimientos?
[6] ¿Las interfaces externas están completamente definidas?
c. Claridad
[¿Existen requerimientos no claros o que necesitan interpretación?]
[7] ¿Eres capaz de entender los requerimientos que están escritos?
(No) (7.a) ¿Las ambigüedades están siendo resueltas satisfactoriamente?
(SI) (7.b) ¿Hay problemas de interpretación o ambigüedad en ellos?
d. Validez
[¿Los requerimientos conducirán al producto que el cliente tiene en mente?]
[8] ¿Existen requerimientos que tal vez no especifiquen lo que el cliente realmente quiere?
(Si) (8.a) ¿Cómo estás resolviendo esto?
[9] ¿Ambas partes están entendiendo lo mismo de los requerimientos?
(Si) (9.a) ¿Tienen un proceso para determinarlo?
[10] ¿Cómo se validan los requerimientos?
• Prototipado

• Análisis
• Simulaciones
e. Factibilidad
[¿Los requerimientos no son factibles desde un punto de vista analítico?]
[11] ¿Hay requerimientos difíciles de implementar técnicamente?
(Si) (11.a) ¿Cuán difícil?
(Si) (11.b) ¿Por qué son difíciles de implementar?
(No) (11.c) ¿Se hizo algún estudio de factibilidad con los requerimientos?
(Si) (11.c.1) ¿Cuan seguro estas de dicho estudio?
f. Precedentes
[¿Los requerimientos especifican algo que no se haya hecho anteriormente, o que la Universidad no haya hecho anteriormente?]
[12] ¿Hay algún estado del arte hecho de los requerimientos?
• Tecnologías
• Métodos
• Lenguajes
• Hardware
(No) (12.a) ¿Algunos son nuevos?
(Si) (12.b) ¿El programa tiene suficiente conocimiento en estas áreas?
(No) (12.b.1) ¿Existe algún plan para adquirir conocimiento en estas áreas?
g. Escala
[Los requerimientos especifican un producto más grande o más complejo o requiere una organización mayor de lo que está acostumbrada la universidad]
[13] ¿El tamaño y complejidad del sistema es una preocupación? (No) (13.a) ¿Se ha hecho algo antes de este tamaño y complejidad? [14] ¿El tamaño del sistema requiere una organización más grande que la acostumbrada?
2. Diseño
a. Funcionalidad
[¿Los problemas potenciales que se discuten son requerimientos funcionales?]
[15] ¿Hay algún algoritmo especificado que tal vez no cumpla con los requerimientos?

(No) (15.a) ¿Los algoritmos y diseño son marginales a los requerimientos?
[16] ¿Cómo determinan la factibilidad de los algoritmos y diseño?
• Prototipando
• Modelando
• Analizando
• Simulando
b. Dificultad
[¿Es difícil de implementar el diseño?]
[17] ¿Algo en el diseño depende de suposiciones irrealistas o optimistas?
[18] ¿Hay algún requerimiento o funcionalidad que es difícil de diseñar?
(No) (18.a) ¿Tienen soluciones para todos los requerimientos?
(Si) (18.b) ¿Cuáles son los requerimientos difíciles de diseñar?
• ¿Por qué son difíciles de diseñar?
c. Interfaces
[¿Las interfaces externas están bien diseñadas y controladas?]
[19] ¿Las interfaces internas están bien diseñadas?
• Software--software
• Software-hardware
[20] ¿Hay algún proceso para definir las interfaces internas?
(Si) (20.a) ¿Hay algún proceso de control de cambios para las interfaces internas?
[21] ¿Hay algún hardware siendo desarrollado paralelamente con el software?
(Si) (21.a) ¿Las especificaciones del hardware están cambiando?
(SI) (21.b) ¿Todas las interfaces hacia el software han sido diseñadas?
(Si) (21.c) ¿Hay algún modelo ingenieril de diseño que puede ser usado para probar el software?
d. Rendimiento
[¿Hay requerimientos rigurosos de tiempo de respuesta o “throughput”?]
[22] ¿Hay problemas con el rendimiento?
• “Throughput”.
• Programación de eventos en tiempo real asíncronos.
• Respuesta en tiempo real.

• Línea de tiempo de recuperación.
• Tiempo de respuesta.
• Tiempo de respuesta de la base de datos, contención o acceso.
[23] ¿Se ha realizado algún análisis de rendimiento?
(Si) (23.a) ¿Cuál es tu nivel de confianza con ese análisis?
(Si) (23.b) ¿Tienen un modelo para tracear el rendimiento desde el diseño hasta la implementación?
e. Capacidad de Probarse
[¿Es el producto difícil o imposible de probar?]
[24] ¿El software será fácil de probar?
[25] ¿El diseño incluye elementos que ayuden a probar el sistema?
[26] ¿El personal de pruebas estuvo involucrado en el análisis de los requerimientos?
f. Restricciones de Hardware.
[¿Hay restricciones ajustadas en el hardware de destino?]
[27] ¿El hardware limita la habilidad de alcanzar algún requerimiento?
• Arquitectura
• Capacidad de memoria
• “Throughput”
• Respuesta en tiempo real
• Líneas de tiempo de recuperación.
• Rendimiento de las bases de datos
• Funcionalidad
• Fiabilidad
• Disponibilidad
g. Software de terceras partes.
[¿Hay algún software siendo usado que no es desarrollado por el equipo?]
Si existe software reutilizado.
[28] ¿Está siendo reutilizado algún software no desarrollado por el equipo?
(Si) (28.a) ¿Se prevé algún problema?
• Documentación

• Rendimiento
• Funcionalidad
• Entrega en tiempo
• Personalización
Si se usa algún software COTS.
[29] ¿Hay algún problema usando el software COTS.
Insuficiente documentación para determinar interfaces, tamaño y rendimiento.
• Bajo rendimiento
• Requiere mucha memoria o espacio de almacenamiento.
• Difícil de unir con el software.
• No suficientemente probado
• Tiene bugs
• No está adecuadamente mantenido.
• Baja respuesta del vendedor
[30] ¿Se prevé algún problema interactuando con las actualizaciones y revisiones del COTS?
3. Código y Unidades de Prueba
a. Factibilidad
[¿La implementación del diseño es difícil o imposible?]
[31] ¿Alguna parte de la implementación no está completamente definida por el diseño especificado?
[32] ¿Los algoritmos y diseño seleccionado son fáciles de implementar?
b. Pruebas
[Los niveles y tiempo asignados a las pruebas de unidad son adecuados?]
[33] ¿Se comenzaron las pruebas de unidad antes de verificar el código respecto al diseño?
[34] ¿Hay suficientes unidades de pruebas especificadas?
[35] ¿Hay tiempo suficiente para realizar todas las unidades de pruebas que se consideran deben realizarse?
[36] ¿Los compromisos se harán respecto a las unidades de prueba si hay problemas con el cronograma?
c. Implementación

[¿Hay algún problema con la implementación?]
[37] ¿Las especificaciones del diseño son suficientemente detalladas para escribir el código?
[38] ¿El diseño está cambiando mientras se codifica?
[39] ¿Hay restricciones del sistema que hacen el código difícil de escribir?
• Tiempo
• Memoria
• Almacenamiento externo
[40] ¿El lenguaje de programación es apropiado para producir el software?
[41] ¿Se están utilizando múltiples lenguajes de programación?
[42] ¿La computadora de desarrollo tiene características de software similares a la de destino? (No) (42.a) ¿Hay diferencias de compilador entre las dos?
4. Integración y pruebas.
a. Ambiente
[¿El ambiente de integración y pruebas es adecuado?]
[45] ¿Hay hardware suficiente para realizar pruebas de integración adecuadas?
[46] ¿Hay algún problema desarrollando escenarios realistas y datos de pruebas para demostrar algún requerimiento?
• Tráfico de datos específicos
• Respuesta en tempo real
• Manejo de eventos asíncronos.
• Actividad multiusuario.
[47] ¿Es posible verificar el rendimiento desde sus instalaciones?
[48] ¿La instrumentación de hardware y software facilitara las pruebas?
(Si) (48.a) ¿Es suficiente para todas las pruebas?
b. Producto
[¿No son adecuadas las definiciones de las interfaces, las instalaciones o el tiempo?]
[49] ¿El hardware de destino estará disponible cuando se necesite?
[50] ¿Se ha acordado un criterio de aceptación para todos los requerimientos?
(Si) (50.a) ¿Hay algún acuerdo formal?
[51] ¿Las interfaces externas están definidas y documentadas?

[52] ¿Algunos requerimientos son difíciles de probar?
[53] ¿Se ha especificado suficiente la integración del producto?
[54] ¿Se ha destinado un tiempo adecuado para la integración y pruebas?
Si COTS
[55] ¿Serán aceptados los datos del vendedor en la verificación de los requerimientos asignados a los COTS?
(Si) (55.a) ¿Está en el contrato?
c. Sistema
[¿Descoordinada integración del sistema, pobre definición de interfaces o instalaciones inadecuadas?]
[56] ¿Se ha especificado suficientemente la integración del sistema?
[57] ¿Se ha asignado un tiempo adecuado para la integración del sistema y las pruebas?
[58] ¿Todos los contratistas forman parte del equipo de integración?
[59] ¿El producto será integrado en un sistema existente?
(No) (59.a.1) ¿Cómo se está garantizando que el producto funcionara correctamente en el sistema existente cuando sea integrado?
[60] ¿La integración del sistema ocurrirá en la parte cliente[5]?
5. Especialidades ingenieriles
a. Mantenimiento
[¿La implementación será difícil de entender o mantener?]
[61] ¿La arquitectura, diseño o código crean dificultades para el mantenimiento?
[62] ¿El personal de mantenimiento estuvo involucrado en el diseño?
[63] ¿La documentación del producto es adecuada para que el producto sea mantenido por alguien diferente a quienes lo construyeron?
b. Fiabilidad
[¿La fiabilidad o disponibilidad son requerimientos difíciles de alcanzar?]
[64] ¿Los requerimientos de fiabilidad están asignados en el software?
[65] ¿Los requerimientos de disponibilidad están asignados al software?
(Si) (65.a) ¿Hay algún problema con los tiempos de respuesta?

[5] No confundir “cliente” con el termino de arquitectura cliente-servidor, se refiere al cliente del proyecto.

c. Seguridad
[¿Hay requerimientos de seguridad no factibles o no demostrables?]
[66] ¿Hay requerimientos de seguridad asociados al software?
(Si) (66.a) ¿Hay alguna dificultad en alcanzar esos requerimientos?
[67] ¿Será difícil de verificar la satisfacción de los requerimientos de seguridad?
[¿Los requerimientos de seguridad son más rigurosos que los acostumbrados?]
[68] ¿Hay requerimientos de seguridad sin precedentes?
[69] ¿Es un sistema "Orange Book"?
[70] ¿Se ha implementado este niel de seguridad anteriormente?
e. Factores Humanos
[¿El sistema será difícil de utilizar debido a una pobre definición de interfaz humana?]
[71] ¿Hay alguna dificultad en alcanzar los requerimientos de factores humanos?
(No) (71.a) ¿Cómo se está asegurando que se lograrán dichos requerimientos?
Si Prototipado:
(Si) (71.a.1) ¿El prototipo morirá?
(No) (71.a.1a) ¿Se está hacienda un desarrollo evolutivo?
(Si) (71.a.1a.1) ¿Se está experimentado con este tipo de desarrollo?
(Si) (71.a.1a.2) ¿Las versiones interinas son entregables?
(Yes) (71.a.1a.3) ¿Esto complica el control de cambios?
f. Especificaciones
[¿La documentación es adecuada para diseñar, implementar y probar el sistema?]
[72] ¿Las especificación de requerimientos es adecuada para diseñar el sistema?
[73] ¿Las especificaciones de hardware son adecuadas para diseñar e implementar el software?
[74] ¿Los requerimientos de interfaces externas están bien especificados?
[75] ¿Las especificaciones de pruebas son adecuadas para probar totalmente el sistema?
Si se está en la fase de implementación o posterior.
[76] ¿Las especificaciones de diseño son adecuadas para implementar el sistema?
1 B. Ambiente de Desarrollo
1. Procesos de desarrollo
a. Formalidad

[¿La implementación será difícil de entender y mantener?]
[77] ¿Se está usando más de un modelo de desarrollo?
• Espiral
• Cascada
• Incremental
(Si) (77.a) ¿La coordinación entre ellos está siendo un problema?
[78] ¿Hay planes controlados y formales para todas las actividades de desarrollo?
• Análisis de requerimientos.
• Diseño
• Codificación
• Integración y pruebas.
• Instalación
• Aseguramiento de la calidad
• Gestión de la configuración
(Si) (78.a) ¿Los planes especifican bien el proceso?
(Si) (78.b) ¿Los desarrolladores están familiarizados con los planes?
b. Adecuación
[¿El proceso es adecuado para el modelo de desarrollo?]
[79] ¿El proceso de desarrollo es adecuado para este producto?
[80] ¿El proceso de desarrollo es soportado por procedimientos, métodos y herramientas compatibles?
c. Control
[¿El proceso de desarrollo es controlado y monitoreado usando métricas? ¿Los sitios de desarrollo distribuidos están coordinados?]
[81] ¿Todos siguen el proceso de desarrollo?
(Si) (81.a) ¿Cómo esto es asegurado?
[82] ¿Se puede medir si el desarrollo está alcanzando los objetivos de productividad y calidad?
Si hay sitios de desarrollo distribuidos.
[83] ¿Hay adecuada coordinación entre los sitios de desarrollo distribuidos?
d. Familiaridad
[¿Los miembros del proyecto están experimentados en el uso de los procesos de desarrollo?

¿Estos son entendidos por todos?]
[84] ¿Las personas se sienten cómodas con los procesos de desarrollo?
e. Control del Producto
[¿Hay mecanismos para controlar los cambios en el producto?]
[85] ¿Hay alguna trazabilidad de los requerimientos desde el código hasta los casos de prueba?
[86] ¿El mecanismo de trazabilidad es utilizado para evaluar el impacto de los cambios en los requerimientos?
[87] ¿Hay algún proceso de cambio formal?
(Si) (87.a) ¿Cubre toda la línea base de los requerimientos, diseño, código y documentación?
[88] ¿Los cambios de cualquier nivel son mapeados desde el sistema hasta las pruebas?
[89] ¿Hay algún análisis adecuado cuando un Nuevo requerimiento entra al sistema?
[90] ¿Hay alguna forma de tracear las interfaces?
[91] ¿Los procedimientos y planes de pruebas son actualizados con los cambios en los procesos?
2. Sistema de desarrollo
a. Capacidad
[¿El poder de las estaciones de trabajo es suficiente (memoria, capacidad de procesamiento o almacenamiento)?]
[92] ¿Hay suficientes estaciones de trabajo y la capacidad de procesamiento que cada cual necesita para todo el personal?
[93] ¿Hay capacidad suficiente para solapar las fases como implementación, integración y pruebas?
b. Adecuación
[¿El sistema de desarrollo soporta todas las fases, actividades y funciones?]
[94] ¿El sistema de desarrollo soporta todas las fases del programa?
• Análisis de los requerimientos.
• Análisis de rendimiento.
• Diseño
• Implementación
• Pruebas
• Documentación

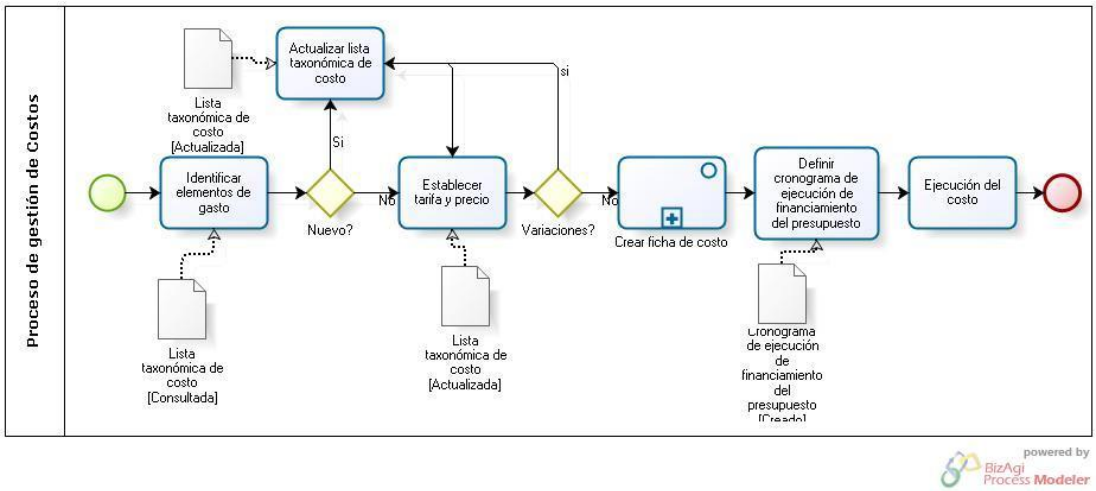
• Gestión de la configuración
• Gestión de proyecto
• Traceabilidad de los requerimientos
c. Usabilidad
[¿Es fácil de usar el sistema de desarrollo?]
[95] ¿Las personas encuentran el sistema de desarrollo fácil de usar?
[96] ¿Hay Buena documentación sobre el sistema de desarrollo?
d. Familiaridad
[¿Hay poca experiencia con el sistema de desarrollo?]
[97] ¿Se han usado anteriormente estas herramientas y métodos?
e. Fiabilidad
[¿El sistema tiene de errores, se reinicia o tiene problemas?]
[98] ¿El sistema es considerado fiable?
• Compilador
• Herramientas de desarrollo
• Hardware
f. Soporte
[¿Hay algún soporte de expertos o del vendedor?]
[99] ¿Las personas están capacitadas en utilizar las herramientas de desarrollo?
[100] ¿Usted tiene acceso a los expertos en utilizar el sistema?
[101] ¿Los vendedores responden a los problemas rápidamente?
g. Entregas
[¿Hay requerimientos de aceptación definidos para entregar el sistema de desarrollo?]
[102] ¿usted es el responsable de entregar el sistema de desarrollo a los clientes?
(Si) (102.a) ¿Existe un presupuesto y planificación adecuado para esta tarea?
3. Procesos de Gestión
a. Planificación
[¿La planificación es oportuna, los temas técnicos están incluidos y la contingencia analizada?]
[103] ¿El programa está siendo dirigido de acuerdo al plan?
(Si) (103.a) ¿Frecuentemente se realizan “maratones” para cumplir los planes?

[104] ¿Se re-planifica cuando ocurren interrupciones?
[105] ¿Las personas de todos los niveles son incluidas en la planificación de su propio trabajo?
[106] ¿Hay planes de contingencia para los riesgos conocidos?
(Si) (106.a) ¿Cómo determinas cuándo activar esos planes?
[107] ¿Los términos a largo plazo son tratados adecuadamente?
b. Organización del Proyecto.
[¿Las relaciones entre los roles están claras?]
[108] ¿La organización es efectiva?
[109] ¿Las personas entienden su rol y el de los otros?
[110] ¿Las personas conocen quien tiene autoridad para qué?

Anexo 44 Vista taxonómica del instrumento de análisis de riegos.

Código	Descripción	Causa	Consecuencia	Fecha de Detección	Fecha de Revisión	Etapa	Tipo de Riesgo	Costo	Acciones de	Costo Mitigación	Acciones de	Responsable	Estado	Observaciones
1														
2														
3														

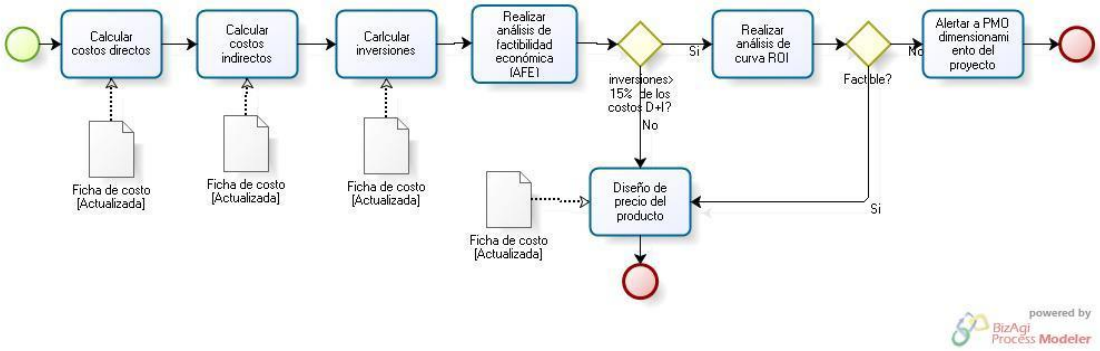
Anexo 45 Proceso de Gestión de Costos



Anexo 46 Tipos de costos

Tipo de costo	Descripción	Ejemplos
Costos Directo	Son aquellos costos que se asigna directamente a una unidad de producción. Por lo general se asimilan a los costos variables. (64)	Mano de obra Capacitación Insumos y materiales
Costos Indirectos	Son aquellos que no se pueden asignar directamente a un producto o servicio, sino que se distribuyen entre las diversas unidades productivas mediante algún criterio de reparto. En la mayoría de los casos los costos indirectos son costos fijos. (64)	Combustible Teléfono Red/conectividad Subcontrataciones Electricidad Agua Servicios Depreciaciones de activos fijos
Inversiones	La inversión es el costo que se encuentra a la espera de la actividad empresarial que permitirá con el transcurso del tiempo, conseguir el objetivo deseado. Tendrán su incidencia en los costos mediante el cálculo de las depreciaciones que se realicen a lo largo de su vida útil. (64)	Tecnología Software Muebles Útiles

Anexo 47 Subproceso Crear Ficha de Costo



Anexo 48 Actividades de los sistemas de trabajo de la PMO

Actividad	Descripción	Frecuencia
Reunión de control y ejecución del presupuesto de la LPS	Supervisa y controla que la LPS cuente con los fondos suficientes y que estos además se ejecuten satisfactoriamente.	15 días
Reunión de chequeo de	Se realiza para controlar que los alcances y	Semanal

cronogramas	tiempos definidos en el cronograma se realicen según las planificaciones pactadas con los clientes.	
Reunión de coordinación del proceso productivo.	Analiza los riesgos globales. Coordina necesidades de comunicación, trabajo en equipo, retroalimentación de conocimiento, etc. Mide resultados para ajuste de métricas. Orienta actividades administrativas.	Semanal
Revisiones técnicas formales de gestión de proyectos	Verifica que todos los procesos de gestión de proyecto funcionen correctamente.	Mensual, por línea de producto
Evaluación de desempeño	Realizada fundamentalmente a los equipos de trabajo buscando competitividad entre ellos, o sea emulando los resultados. También se realiza para los jefes de equipo buscando elevar la preparación de los cuadros.	Mensual o trimestral

Anexo 49 Actividades del sistema de trabajo del equipo de proyecto usando SCRUM

Actividad	**Descripción**	**Involucrados**	**Tiempo estimado de duración**
Reunión de definición del Product Backlog	Se define y detalla la lista de objetivos o requisitos priorizados del producto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. (52)	SCRUM Master Especialista funcional SCRUM Owner (clientes)	4 horas
Planificación de la iteración (Sprint Planning)	Se realiza respecto a dos objetivos fundamentales: **Selección de requisitos** (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita. **Planificación de la iteración** (4 horas máximo). El equipo elabora la lista de tareas de la	SCRUM Master Especialista funcional SCRUM Owner (clientes) SCRUM Team	8 horas

	iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas. (52)		
Reunión diaria de sincronización del equipo (SCRUM daily meeting)	El objetivo de esta reunión es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad, al poner de manifiesto puntos en que se pueden ayudar unos a otros. Cada miembro del equipo debe responder las siguientes preguntas: ¿Qué he hecho desde la última reunión de sincronización? ¿Pude hacer todo lo que tenía planeado? ¿Cuál fue el problema? ¿Qué voy a hacer a partir de este momento? ¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto? (52)	SCRUM Master SCRUM Team	15 minutos
Demostración de requisitos completados (Sprint Review)	Reunión informal donde el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo, haciendo un recorrido por ellos lo más real y cercano posible al objetivo que se pretende cubrir. (52)	SCRUM Master SCRUM Owner (clientes) SCRUM Team	4 horas
Retrospectiva (Sprint Retrospective)	El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de	SCRUM Master SCRUM Team	horas

	manera continua su productividad. (52)		

Anexo 50 Indicadores del informe de monitoreo de la PMO

1. Planificación
1.1. Cantidad de tareas planificadas.
1.2. Cantidad de tareas atrasadas.
1.3. Variaciones del cronograma respecto a la línea base inicial y la última corrección realizada.
1.4. Índice de Rendimiento del Cronograma (SPI) (55).
1. Recursos Humanos
1.1. Utilización de las fuerzas laborales = Cantidad de tareas / Hombres
1.2. Eficiencia de utilización de los recursos humanos = cantidad horas hombres planificadas/Hombres.
2. Riesgo
2.1. Principales riesgos identificados.
2.2. Principales incidencias.
3. Costo
3.1. Estado de ejecución del presupuesto.
3.2. Índice de Rendimiento del Coste (CPI) (55)
3.3. Volumen de sobre giro o ahorro del presupuesto inicial
4. Calidad
4.1. Cantidad de no conformidades reportadas
4.2. Cantidad de no conformidades pendientes
4.3. Cantidad de no conformidades en ejecución
4.4. Cantidad de no conformidades en resueltas
5. Variación
5.1. Cantidad de solicitudes de cambios reportadas
5.2. Cantidad de solicitudes de cambios ejecución
5.3. Cantidad de solicitudes de cambios pendientes
5.4. Cantidad de solicitudes de cambios realizadas

Anexo 51 Indicadores del informe resultado de la PMO

1. Plan de producción
1.1. Área de dominio (Definido por los elementos definidos en la arquitectura de dominio)
1.1.1.Cantidad de activos de software planificados.
1.1.2.Cantidad de activos de software terminados.
1.1.3.Porciento de cumplimiento del plan.
1.2. Área de aplicaciones (Definido por el volumen de contratos realizados)
1.2.1.Cantidad de productos planificados.
1.2.2.Cantidad de productos terminados.
1.2.3.Por cada productos Porciento de terminación.
1.2.4.Porciento de cumplimiento del plan.
2. Productividad
2.1. Índice de utilidades alcanzado.
2.2. Índice de productividad por hombre.

2.3. Salario medio productividad.
2.4. Reutilización
2.4.1. Cantidad de elementos reutilizados, históricos y del mes (Numero de activos de software reutilizados * productos de clientes terminados)
2.4.2. Índice de reutilización, histórico y del mes (Cantidad de componentes de los productos de clientes/cantidad de componentes reutilizados en los productos de clientes) Cuando índice de reutilización tiende a uno existen los mejores resultados.
2.4.3. Componentes más reutilizados, históricos del mes.
3. Ciclo de producción.
3.1. Tiempo promedio histórico de soluciones de software que responden a contratos de clientes.
3.2. Tiempo promedio de duración de productos terminados en el mes que responden a contratos de clientes.
4. Calidad
4.1. Productos
4.1.1. Cantidad de no conformidades reportadas de clientes en el mes.
4.1.2. Índice de errores detectados = cantidad de no conformidades / cantidad de productos desplegados.
4.1.3. Cantidad de clientes satisfechos
4.2. Proceso
4.2.1. Cantidad de revisiones técnicas formales realizadas.
4.2.2. Cantidad de no conformidades del proceso detectadas.
4.2.3. Índice de calidad del proceso (Por persona y por equipos) = Cantidad de no conformidades del proceso/ Cantidad de hombres involucrados.

Anexo 52 Activos de la LPS GDS

LPS GDS
• Componente para la gestión de calendarios v1.0
• Componente para la gestión de estructuras arboleas v1.3
• Componente para la gestión de localidades y direcciones v1.0
• Componente para la gestión de caja v1.0
• Componente de puntos de ventas v1.4
• Componentes para la gestión de tarifas y sistemas de precio v1.0
• Componente para la definición dinámica de conceptos de ventas y medios de transporte v2.1
• Componente para la gestión de planos v2.1
• Componentes para la gestión de boletines y sus formatos v1.2
• Componentes para la gestión de ambiente comercial v1.1
• Componente para la facturación de servicios v 1.6

Anexo 53 Activos de la LPS paquete de gestión administrativa del transporte

LPS Paquete de soluciones de gestión administrativa orientada a la actividad del transporte
• Componente para la gestión de eventos v1.3
• Componente para la gestión de noticias v1.1
• Componente chat corporativo v1.5
• Componente gestión de personas v2.1
• Componentes para la gestión de tarjetas v2.2

• Componente para la gestión de medios de transporte v2.1
• Componente para la gestión control de partes e inventarios de medios de transporte v2.3
• Componente para la gestión de incidencias v1.2
• Componente para la gestión de archivo digital v3.0
• Componente para la gestión dinámica de indicadores v1.1
• Componente para el control de operaciones y su formalización en registro contables v2.4

Anexo 54 Activos de la LPS Sistema de Gestión Bancario

LPS Sistema de gestión bancario
• Planificador dinámico v1.0
• Componente para la gestión de localidades y direcciones v1.0
• Componente de gestión de multientidad v1.0
• Componente de gestión de tareas programadas v1.3
• Componentes para la gestión de servidor de Fechas v1.0

Anexo 55 Activos de la LPS Sistemas de Gestión Administrativa

LPS Sistemas de gestión administrativa
• Componente para la gestión de MultiEntidad v1.0
• Componente para la gestión dinámica de nomencladores v2.1
• Componente para la gestión de conexiones y el despliegue v1.5
• Componente de edición y generación de reportes v3.2
• Componente para la gestión de localidades y direcciones v1.0
• Componente gestión contable v3.4
• Componente gestión de facturas v2.1
• Componente gestión de almacén v2.3
• Componente gestión de nomina v2.0

Anexo 56 Especificación catalogo componentes

Catálogo de componentes
1. Componente
• Descripción
• Fecha entrada
• Responsabilidad arquitectónica
• Categoría tecnológica
• Categoría temática
• Procesos que cubre
• Versión
• Tecnología base
• Responsable
• Registro de salidas
a. Fecha salida
b. Destino
c. Valor comercial
• Cantidad de reutilizaciones

Anexo 57 Registro de versión de despliegue

1. Registro de versión de despliegue

a. Producto
• Despliegue
• Versión
• Cliente
• Fecha despliegue
• Condición comercial
• Línea base del producto
• Encargado del despliegue
• Duración del despliegue

Anexo 58 Especificación técnica del producto

1. Especificación técnica del producto
• Producto
• Código producto
• Versión
• Requerimientos mínimos de hardware
• Requerimientos mínimos de ambiente (red, servidores, almacenamiento, ect)
• Componentes
a. Especificar cada componente según Anexo 56
• Manual de usuario (a considerar)

Anexo 59 Especificación del expediente documental propuesto

• LPS
○ Calidad
▪ Arquitectura
• art_Lista de chequeo de la arquitectura
• art_Lista de chequeo de especificación de componente
▪ Planificación
• art_Lista de chequeo sistema de trabajo PMO y Equipos de Trabajo
• art_Lista de chequeo planificación
▪ Riesgos
• art_Lista taxonómica de riesgos
• art_Cuestionario de identificación de riesgos
▪ Costos
• art_Taxonomía de gastos
○ Concepción
▪ art_Proyecto técnico
▪ art_Contrato
▪ art_Acta de denegación
▪ art_Estudio de factibilidad
○ Arquitectura
▪ art_Arquetipo Dominio
• comp_Sistema
• comp_Información
• comp_Infraestructura
• comp_Tecnología

- Repositorio
  - Activos
    - art_Catálogo de activos
      - comp_Activos
        - comp_temÁticos
        - comp_tecnolÓgicos
        - comp_Otros
  - Familia de producto
    - art_Registro de versión de despliegue
  - Solicitudes de entrada
    - art_Solicitud de entrada 1
    - art_Solicitud de entrada 2
  - Solicitudes de salida
    - art_Solicitud de salida 1
    - art_Solicitud de salida 2
- Unidad de negocio(Línea de producto-Subsistema)
  - Ingeniería
    - Modulo
      - Componente
        - art_Especificación de componente
        - art_Manual de usuario
          - comp_usuario
          - comp_instalación
          - comp_configuración
        - art_Caso de estudio
        - Ordenes de mantenimiento
          - art_Orden mantenimiento 1
          - art_Orden mantenimiento 2
  - Calidad
    - Modulo
      - Componente
        - No conformidades
          - art_No conformidad
        - Solicitudes de cambios
          - art_Solicitud de cambio
        - Casos prueba
          - art_Caso prueba 1
          - art_Caso prueba 2
  - Gestión
    - Riesgos
      - art_Instrumento de gestión de riesgos
      - art_Acta periódica semanal
    - Costo
      - art_Instrumento de gestión de costo o ficha de costo.
      - art_Cronograma de ejecución financiero y presupuesto.

- Planificación
  - art_Registro de variaciones de planificación
  - art_Producto Backlog
    - comp_Sprint 1
    - comp_Sprint 2
- Organización
  - Acuerdos
    - art_Registro de acuerdos
  - Actas
    - art_Acta de reunión
- Seguimiento y Control
  - Informes de monitoreo
    - Semena N
      - art_Informe de Monitoreo
  - Informes de resultado
    - Mes N
      - art_Informes de resultado

- Familias de producto
  - Producto
- Concepción
  - art_Proyecto técnico
  - art_Contrato
  - art_Acta de denegación
  - art_Estudio de factibilidad
- Ingeniería
  - Arquitectura
    - art_Arquitectura de aplicaciones
      - comp_Instancia Arquetipo de Dominio
      - comp_Registro de Variabilidad
- Calidad
  - No conformidades
    - art_No conformidad
  - Solicitudes de cambios
    - art_Solicitud de cambio
  - Casos prueba
    - art_Caso prueba 1
    - art_Caso prueba 2
  - Apta de aceptación
- Gestión
  - Riesgos
    - art_Instrumento de gestión de riesgos
    - art_Acta periódica semanal
  - Costo
    - art_Instrumento de gestión de costo o ficha de costo.

• art_Cronograma de ejecución financiero y presupuesto.
o Planificación
▪ art_Registro de variaciones
▪ art_Producto Backlog
• comp_Sprint 1
• comp_Sprint 2
o Organización
▪ Acuerdos
• art_Registro de acuerdos
▪ Actas
• art_Acta de reunión
o Seguimiento y Control
▪ Informes de monitoreo
• Semena N
o art_Informe de Monitoreo
▪ Informes de resultado
• Mes N
o art_Informe de resultado
• Empaquetado y Despliegue
o art_Plan de entrenamiento
o art_Plan de adquisición
o art_Caso de estudio
o art_Manual de usuario
o art_Especificación técnica del producto

Anexo 60 Ejemplo de aplicación del patrón de versionado propuesto.

Tómese como Línea Base matriz la Arquitectura de Dominio de un entorno de negocio de gestión hospitalaria.

Una simplificada representación de la Arquitectura de Dominio de este entorno identifica como activos de software de la Lnea Base matriz 6 componentes temáticos especializados.

LBM->Línea Base matriz (entorno de negocio Gestión Hospitalaria) Versión 1.0.3 (el 1 representa la Línea Base matriz del entorno de negocio, el 0 es debido a que las LBM no presentan variaciones en este estado, ya que todos sus elementos se encuentran en la fase de Ingeniería de Dominios, el 3 representa la tercera iteración que sufrió en calidad la misma, para cumplimentar el flujo de liberación integral completo de la LBM), primer nivel de abstracción de la política de versionado.

Componente 1 -> Admisión. Versión 1.0.2 (el uno significa que el componente está en su versión matriz 1, el cero que no ha sufrido variación y el 2 que se necesitaron dos iteraciones para su liberación de calidad), tercer nivel de abstracción de la política de versionado.

Componente 2 ->Imageneología. Versión 1.0.7 (el uno significa que el componente está en su versión matriz 1, el cero que no ha sufrido variación y el 7 que se necesitaron siete iteraciones para su liberación de calidad), tercer nivel de abstracción de la política de versionado. Así

sucesivamente corresponde con el resto de los componentes ejemplificados

- Componente 3 -> Archivo. Versión 1.0.6
- Componente 4 -> Consulta. Versión 1.0.4
- Componente 5 -> Laboratorio. Versión 1.0.1
- Componente 6 -> Ingreso. Versión 1.0.1
- Componente 7 ->Gestión clínica. Versión 1.0.8

Ahora suponiéndose que aparece como cliente especifico hipotético de nuestra fábrica de software el "Hospital MMM23". Ante un estudio de factibilidad y la aplicación del arquetipo arquitectónico contenido en la LBM 1.0.3, realizado el equipo de arquitectura de la fábrica de software, se identifican las siguientes variaciones evolucionándose a la Línea Base ramificada LBR versión 2.2.5 que queda conformada por los siguientes elementos:

LBR->Línea Base aplicación (cliente concreto Hospital MMM23) Versión 2.2.5 (el 2 representa la Línea Base aplicación del Hospital MMM23 y significa que es la primera derivación desarrollada en la fábrica de la LBM 1.0.3, el segundo 2 representa el número de componentes incorporados a la Línea Base de aplicación y el 5 representa las cinco iteraciones que sufrió en calidad la LBR para cumplimentar el flujo de liberación integral completo del producto), **primer nivel de abstracción de la política de versionado.**

- Componente 1 -> Admisión. Versión 2.1.3 (el 2 significa en este caso que el componente ha sido ramificado de su versión original de la LBM 1 evolucionando a la LBM versión 2., el 1 que es la primera variación sufrida de su componente base y el 3 que se necesitaron tres iteraciones para su liberación final de calidad), tercer nivel de abstracción de la política de versionado.
- Componente 3 -> Archivo. Versión 2.1.5 (el 2 significa en este caso que el componente ha sido ramificado de su versión original de la LBM 1 evolucionando a la LBM versión 2., el 1 que es la primera variación sufrida de su componente base y el 5 que se necesitaron cinco iteraciones para su liberación final de calidad), **tercer nivel de abstracción de la política de versionado.**
- Componente 4 -> Consulta. Versión 1.0.4
- Componente 5 -> Laboratorio. Versión 1.0.1
- Componente 6 -> Ingreso. Versión 1.0.1
- Componente 7 -> Gestión clínica. Versión 2.1.2

Como se puede observar en esta LBR fue omitido por no aplicar en el arquetipo aplicado al Hospital MMM23 el componente 2, que representa el área de proceso de imageniología, puesto que este hospital no cuenta con este servicio.

Componente 8 -> Consulta especializada. Versión 2.0.2 (el 2 significa en este caso que el componente ha sido incorporado la LBR 2, por tanto asume el número de versión de su Línea Base padre, el 0 que el mismo no ha sufrido variación aun de su versión original y el segundo 2 que se necesitaron dos iteraciones para su liberación final de calidad), **tercer nivel de abstracción de la**

política de versionado

Componente 9 -> Base de conocimiento de inferencia de diagnostico. Versión 2.0.5

Anexo 61 Por ciento de cubrimiento del modelo propuesto por las áreas prácticas del modelo del SEI

Ingeniería de Software	**%Cubre**	**Gestión Técnica**	**%Cubre**	**Gestión Organizacional**	**%Cubre**
Definición de la arquitectura	100%	Gestión de la configuración		Construcción de casos de estudio del negocio	50%
Evaluación de la arquitectura		Análisis de factibilidad	90%	Gestión de clientes	
Desarrollo de componentes	100%	Seguimiento y control	90%	Gestión de adquisición	10%
Minería de activos existentes	100%	Definición del proceso	100%	Gestión de costos	90%
Ingeniería de requisitos	90%	Gestión del alcance	100%	Implantación y soporte	90%
Integración de sistemas	100%	Planificación técnica	100%	Análisis de mercado	10%
Pruebas	100%	Gestión de riesgos técnicos	100%	Operaciones	
Comprensión de Dominios relacionados	90%	Herramientas de soporte		Planificación de la organización	
Reutilización de software de terceros				Gestión de riesgos de la organización	
				Estructuración de la organización	20%
				Observatorio tecnológico	
				Entrenamiento	10%