Hacia una Línea de Productos – Diseño de Arquitecturas de Referencia en el Dominio de la Computación Móvil

Francisca Losavio,[i] Oscar Ordaz[ii] & María Dolores Nardi[iii]

Resumen

Las *Líneas de Productos de Software (LPS)* basadas en la reutilización, pretenden mejorar tiempo de comercialización, evolución del software y bajar costos de desarrollo. Productos de software, miembros de la familia LPS, son derivados instanciando una *Arquitectura de Referencia (RA: "Reference Architecture")* genérica, con componentes comunes y variantes. La ventaja para una empresa es obtener productos concretos instanciando la RA, reutilizando un repositorio de activos (*"assets")*. Sin embargo, diseñar una RA para un dominio dado, es tarea compleja y costosa en vista de que ésta debe ser evolutiva, acorde con los cambios tecnológicos. Se puede atacar el problema considerando que la empresa crea desde cero su LPS, aplicando una estrategia descendente *("top-down")*, para definir el repositorio de activos inspirándose en productos del mercado existentes en el dominio. Otra forma es seguir una estrategia ascendente (*"bottom-up")* cuando una empresa tiene varios productos desarrollados y determinar mediante un proceso de reingeniería, componentes o módulos que pueden ser reutilizados como activos; esta es la estrategia seguida en la presente investigación. El objetivo de este trabajo es describir un proceso bottom-up semi-formal, sistemático y repetible de diseño de la RA, representada por un grafo conexo no dirigido. El proceso es aplicado al caso de estudio de aplicaciones de Transacciones Financieras, sub-dominio de la Computación Móvil del consorcio *CONECTIUM Limited C.A*., que también contempla aplicaciones en los sub-dominios Sistemas de Salud y Contenidos de Entretenimiento. Los resultados obtenidos son ilustrativos, repetibles y de utilidad para la práctica industrial, facilitando así la Ingeniería de una LPS.

Palabras clave: línea de productos de software, arquitectura de referencia,

[iii] Universidad Central de Venezuela – Venezuela. Coordinadora del Laboratorio de Modelos, Software y Tecnologías (MoST) de la Escuela de Computación.

[ii] Universidad Central de Venezuela – Venezuela. Miembro del Laboratorio de Modelos, Software y Tecnologías (MoST)

[iii] CONECTIUM LIMITED C.A. Venezuela.

diseño de una arquitectura de referencia, proceso bottom-up, computación móvil, ingeniería del dominio

Abstract

Software Product Lines (SPL), based on reuse, claim to improve software evolution, time to market and decrease software development costs. Concrete software products, members of the SPL family, are derived by instantiating a generic *Reference Architecture (RA)*, holding common and variant components. Advantages for an enterprise are to obtain concrete products derived from RA by reusing the asset repository, which has been built in parallel while constructing RA. However, the RA design is a complex and costly task, since it must be evolutionary according to technological changes. A way to cope with this problem is to consider a top-down strategy, where the enterprise builds the SPL from zero; however existing products on the market must be studies to construct the asset repository. Another way is to follow a bottom-up strategy, considering products already developed by the enterprise and apply a reengineering process to determine components or modules that can be reused as assets; this is the strategy followed in this research. The goal of this work is to describe a systematic and repeatable bottom-up process to design RA, represented by a non-directed and connected graph. The process is applied to the case study of Finantial Transactions applications, sub-domain of the Mobile Computing domain of the *CONECTIUM Limited C.**A*** consortium, holding also sub-domains of Healthcare and Entretainment Contents applications. It is expected that the results obtained will be illustrative, repeatable and useful for the industrial practice, to facilitate SPL Engineering.

Keywords: software product line, reference architecture, design of reference architecture, bottom-up process, mobile computing, domain engineering

--

Fecha de recepción 10/02/2017 - Fecha de aceptación 04/08/2017

I. Introducción

Hace más de una década surgió en *Ingeniería del Software (IS)* la tendencia al desarrollo basado en componentes reutilizables, para configurar nuevos sistemas de software complejos con más rapidez, mayor calidad y a más bajo costo. El desarrollo a gran escala de estos sistemas en general intensivos, caracterizados por su complejidad, representa un desafío para los investigadores, líderes de proyectos y desarrolladores. El enfoque de *Líneas de Productos de Software (LPS),* conocido también en la literatura como *"SPL: Software Product Lines"*, o simplemente *Líneas de Productos* o *"PL: Product Lines"* [1] [2] [3] [20], permite la construcción de soluciones individuales basadas en un repositorio de componentes de software reutilizables comunes y variables. El hecho de proporcionar soluciones individuales responde a diversas necesidades en el software con respecto a la funcionalidad, la tecnología subyacente y las propiedades no funcionales, como por ejemplo rendimiento y espacio de memoria. Las LPS prometen distintos beneficios, de los cuales los más importantes son: adaptabilidad a las necesidades del cliente, reducción de costos, mejora de la calidad global, evolución, disminución del tiempo de comercialización y mantenimiento de las aplicaciones posterior a su entrada en producción. Este enfoque representa un gran reto, principalmente por la tarea de construir artefactos denominados *activos o "assets"*, suficientemente genéricos para poder ser reutilizados (interoperabilidad de código, etc.) en la configuración de nuevos sistemas, acordes con las necesidades de los clientes o del negocio. La *Ingeniería de Líneas de Productos de Software (ILPS),* definida en el marco de referencia estándar ISO/IEC 26550 [3], consta de dos ciclos de vida principales: *Ingeniería del Dominio (ID)* e *Ingeniería de la Aplicación (IA)*.

La ID abarca el análisis y la identificación del ámbito de aplicación de la LPS, que incluye la captura de todo el conocimiento sobre el dominio y el negocio, a través del modelado de *partes comunes* o *"commonality"* y de partes variables, los *puntos de variación o "variation points"* [20], que están presentes en la *Arquitectura de Referencia (o RA: Reference Architecture)*, la cual es una arquitectura de software de alto nivel de abstracción, genérica y basada en un esquema instanciable o *modelo de variabilidad*, que dirige la derivación de productos concretos de la familia LPS. Una *arquitectura de software,* cuya forma o topología se denomina *configuración arquitectónica*, es un conjunto de componentes de software (sistemas, módulos, paquetes, programas, etc.), relacionados entre sí por conectores y que ofrece un cierto comportamiento [4]; en nuestro caso se representa por un grafo conexo no dirigido y la vista lógica de su configuración se especificará en UML 2.0 [5]. Un *estilo arquitectónico* define una familia de sistemas en términos de un patrón de la organización estructural o topología de los componentes y conectores.

La IA concierne la generación de productos concretos a partir de la RA, es decir su objetivo es definir las estrategias de reutilización de los activos de acuerdo

a los requisitos específicos del cliente. Un producto concreto de la familia LPS, derivado de la RA, debe poseer una configuración arquitectónica "válida", es decir conexa (no tener componentes aislados), consistente (cumplir con las restricciones especificadas) y funcionalmente idónea (cumplir con todas las tareas para las cuales fue diseñada [6], implicando esto la satisfacción de *requisitos funcionales (RF) y no funcionales (RNF)*. En general, el repositorio de activos debe contener "grosso modo" los componentes arquitectónicos diseñados a un alto nivel de abstracción en la RA, así como las conexiones con los respectivos módulos de bajo nivel de abstracción (código fuente), para así configurar el código ejecutable del producto en el momento de la derivación.

En el área de la IS, a raíz de la rápida conexión en red a nivel mundial que ha originado la "globalización", surge recientemente la *Computación Móvil* [7] [8], la cual es una tendencia de desarrollo de software basado en Tecnologías Informáticas de Comunicación (TIC) que permiten transmisión multimedia (datos, voz, video) vía un computador o cualquier otro dispositivo móvil con conexión inalámbrica.[1] En esta área, emerge la disciplina *Ingeniería de Software Móvil (ISM)* [8]; los principales conceptos en ISM son la comunicación móvil, el hardware móvil y el software móvil. En particular, el *software móvil* se caracteriza por: - *aplicaciones Web*, denominados sitios Web o "*Web Apps*" que despliegan páginas vía un browser, - las denominadas *aplicaciones nativas* o "*Apps*" que son desarrolladas separadamente para un dispositivo particular y son descargadas e instaladas por el usuario, y - las *aplicaciones híbridas* como una manera de exponer contenido de una Web App en el formato y dispositivo de una App. El software móvil no se compone de sistemas muy complejos o intensivos que involucran muchas miles de líneas de código, pero debe ser desarrollado con gran rapidez para satisfacer la demanda creciente del mercado y su naturaleza dinámica, pero además deben ser de calidad, porque este factor influencia su aceptación o rechazo inmediato por parte de un amplio público de usuarios; el uso de TIC de rápida implementación y evolución caracteriza este tipo de desarrollo. La satisfacción del cliente es el principal objetivo de la ISM, como en general se espera de cualquier disciplina de ingeniería. El éxito de una aplicación, sea ésta móvil o no, depende no solo del cumplimiento adecuado de su funcionalidad o idoneidad funcional, sino también de la satisfacción de las propiedades de calidad requeridas por esa funcionalidad [3]; por abuso de lenguaje, las propiedades de calidad a veces son denominadas directamente RNF en la literatura [8] [9]. Nuestro enfoque destaca la identificación de estas propiedades de calidad y su clara trazabilidad respecto a los RF, para así garantizar una RA evolutiva en el dominio de la Computación Móvil caracterizada por su dinamicidad. Las propiedades de calidad son en gran parte responsables de

[1] http://www.tutorialspoint.com/mobile_computing/mobile_computing_overview.htm

la evolución de la RA en el tiempo. En nuestro contexto, un *dominio* es el conjunto mínimo de propiedades que describen con precisión una familia de problemas que involucran en su solución aplicaciones o sistemas computacionales [10]; en pocas palabras, es el tratamiento computacional de un sector del mercado o negocio.

Este trabajo propone como principal objetivo un proceso de *Ingeniería del Dominio de la Computación Móvil (IDCM)*, inspirado en Losavio, Ordaz y Esteller [9] y enfocado a la migración de una empresa de desarrollo de software móvil que puede abarcar varios sub-dominios diferentes, hacia una LPS en el dominio de la computación móvil, con el fin de: a) implementar un primer repositorio de activos con componentes comunes y variables especificados a un alto nivel de abstracción, b) diseñar arquitecturas de referencia para los sub-dominios, mediante un enfoque ascendente o *"bottom-up"* basado en el estudio de productos existentes ya desarrollados en la empresa, c) aplicar el proceso paso a paso a un caso de estudio real, d) definir estrategias para instanciar las RA obtenidas para cada sub-dominio, a partir de los activos identificados como reutilizables.

Se propone en general aplicar buenas prácticas de la IS a la ISM, según se recomienda en [11], las cuales no son muy utilizadas en el dominio emergente de la Computación Móvil [8]. Aspectos importantes a ser considerados relativos a las propiedades de calidad ya mencionadas son: - la integración de software y hardware, - los recursos limitados como almacenamiento y despliegue de la Interfaz Usuario (*UI: User Interface*) en pantallas pequeñas con diferente resolución y en diferentes dispositivos, - cambios frecuentes de tecnología, - uso de estándares para conseguir interoperabilidad de la información y propiedades tradicionales como - portabilidad a diferentes plataformas, - confiabilidad en cuanto a la disponibilidad de la conexión y de la información con replica de datos locales en vista de la posible fallas en las conexiones Internet, - seguridad para control de acceso y - eficiencia en el despliegue de la UI y en la ejecución de tareas funcionales y/o servicios. Los métodos de desarrollo denominados "ágiles" [12] [13] [14], dirigidos tradicionalmente hacia un desarrollo rápido de la UI y con uso de prototipaje para satisfacción del cliente, son recomendados para este tipo de software móvil [8] [15]. Sin embargo estos métodos ligeros no se adaptan al desarrollo de una LPS centrada en requisitos de reutilización, evolución y calidad [16]; ni siquiera los métodos "clásicos" dirigidos al desarrollo de sistemas "tradicionales" complejos, como por ejemplo grandes sistemas de información empresariales, sistemas de control de plantas o sistemas de transporte, se adaptan bien al desarrollo de una LPS y actualmente aún no se han establecido claramente métodos completos adaptados a este enfoque de desarrollo. Una extensa ID es requerida para garantizar los requisitos antes mencionados; las actividades para la adopción y construcción de una LPS no son triviales [3] y contemplan una inversión financiera inicial fuerte y un esfuerzo considerable en tiempo [17] [18] [19].

El presente trabajo se estructura de la siguiente forma, además de esta primera sección con la Introducción: la Sección 2 presenta el contexto y algunos trabajos relacionados directamente con este estudio; la Sección 3 describe el proceso IDCM, configurado para el consorcio CONECTIUM Limited C.A, Venezuela, el cual incluye tres filiales o sub-dominios: *Transacciones Financieras, Sistemas de Salud* y *Contenidos de Entretenimiento*; se contemplan tres fases de la ID: Alcance, Ingeniería de Requisitos del Dominio y Diseño del Dominio. La sección 4 presenta la aplicación del proceso IDCM al caso concreto del consorcio CONECTIUM para el sub-dominio TransaMóvil y algunas recomendaciones generales; también se muestra el Repositorio de Activos obtenido para el consorcio. Finalmente en la Sección 5 se plantean las conclusiones y perspectivas.

II.Contexto

II.1 Ingeniería de Líneas de Productos de Software (ILPS)

El marco de referencia estándar ISO/IEC 26550 [3] para la ILPS que se muestra en la Figura 1, se inspira basicamente en el trabajo de Pohl, Bockle y Van Der Linden [20]. Se distinguen dos ciclos de vida principales: *Ingeniería del Dominio (ID) e Ingeniería de la Aplicación (IA).*

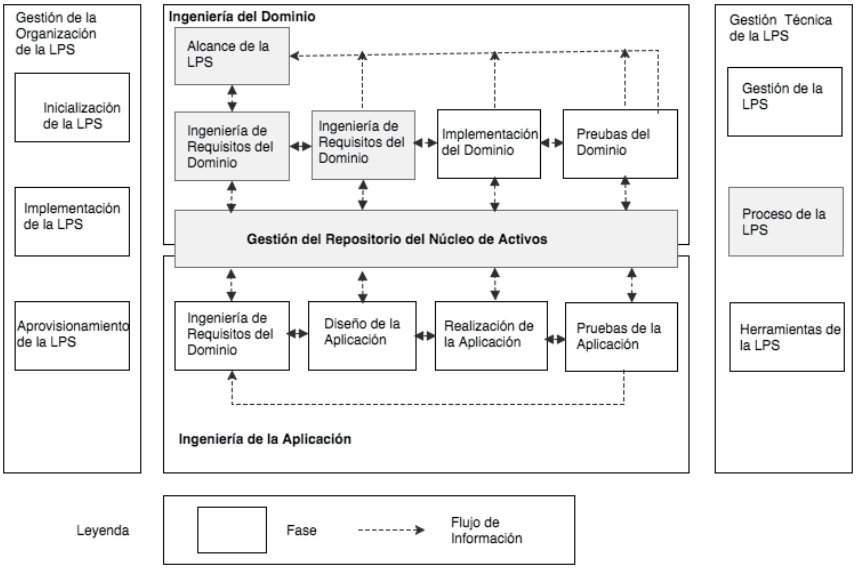

Fig. 1: Modelo de Referencia para Líneas de Productos de Software y Sistemas ISO/IEC 26550. Fuente: [3]; traducido al castellano por los autores.

En nuestro caso estamos interesados en el ciclo ID, el cual incluye las fases *Alcance de la LPS, Ingeniería de Requisitos del Dominio, Diseño del Dominio,* los cuales están resaltadas en gris en la Figura 1 porque serán tratados explícitamente en este trabajo por referirse directamente a la construcción de la RA, junto con la *Gestión del Repositorio del Núcleo de Activos*; además en la ID se contemplan las fases de *Realización del Dominio y Pruebas del Dominio*, que no serán tratadas en este trabajo. Entre los ciclos de vida ID e IA se sitúa la Gestión del Repositorio del Núcleo de Activos, en el cual se ubican los activos obtenidos en todas las fases de la ID y también es realimentado por los activos producidos durante la derivación de los productos concretos obtenidos instanciando la RA en la IA. Como componentes transversales de la ILPS, se mencionan la *Gestión de la Organización de la LPS y la Gestión Técnica de la LPS,* que están involucradas en los dos ciclos del desarrollo. La Gestión Técnica trata en particular de la especificación del *Proceso de la LPS* el cual nos concierne directamente y también es resaltado en gris en la Figura 1, las *Herramientas de Soporte* y en general toda la *Gestión de la LPS*. En nuestro caso enfocamos la especificación del proceso para el ciclo de vida ID, en las fases ya mencionadas de *Alcance, Ingeniería de Requisitos del Dominio y Diseño del Dominio,* que culminan con la construcción de la RA, en paralelo con la construcción del *Repositorio del Núcleo de Activos*.

La ID es el primer ciclo de la ILPS; consiste en el conjunto de actividades cuyo objetivo es desarrollar, mantener y administrar la identificación, especificación y evolución de los dominios [2] [3] [20]. Ha sido de especial interés para la comunidad de IS, por varias razones: a) la necesidad de mantener y reutilizar los conocimientos existentes, b) la necesidad de gestionar la variabilidad creciente de los requisitos de la información que se maneja y de los sistemas de software, y c) la necesidad de obtener, formalizar y compartir conocimientos especializados en diferentes ámbitos, actualmente en rápida evolución. La ID como disciplina tiene una importancia práctica, ya que debe proporcionar los métodos y técnicas que pueden ayudar a reducir el tiempo de entrega de los productos al mercado, los costos de desarrollo y los riesgos en los proyectos de desarrollo de software. La ID es utilizada, investigada y especificada en diversas áreas, principalmente en ILPS, ILED (Ingeniería de Lenguajes Específicos de Dominio) y el Modelado conceptual.

Según Bjørner [21], antes de diseñar el software se deben comprender sus requisitos, y antes de que se puedan desarrollar los requisitos, se debe comprender el dominio de la aplicación que se va a construir [10]; para comprender el domino, se debe realizar la etapa "tradicional" de Análisis del Dominio, bajo la óptica de los grupos de interés o "stakeholders". En ILPS esta etapa puede estar incluida en la fase de Alcance donde se establecen los límites del software que se va a construir y se puede enriquecer luego en la fase Ingeniería de Requisitos del Dominio [3]. Un dominio conlleva características predefinidas, como por ejemplo un estilo

arquitectónico y propiedades de calidad globales a las que debe responder la arquitectura y la familia de sistemas desarrollados bajo ese dominio, los cuales generalmente comparten elementos comunes. En particular, se observa que la etapa de Alcance reduce el esfuerzo de trabajo realizado posteriormente en las fases de Ingeniería de Requisitos y Diseño del Dominio. Todo lo anterior dentro de un marco de trabajo iterativo y relativamente ágil, en permanente revisión y movimiento durante los proyectos hasta llegar al resultado deseado.

II.2 La Computación Móvil

Se ha mencionado que debido a la rápida conexión en red a nivel mundial que ha inducido la "globalización", por la introducción de nuevas TIC, surge la disciplina o dominio de la *Computación Móvil* [7] [8]*,* la cual es una tendencia de desarrollo de software basado en TIC que permiten transmisión multimedia (datos, voz, video) vía Internet, hacia un computador, laptop, tableta o cualquier otro dispositivo móvil con conexión inalámbrica. En particular, todos los dispositivos móviles tienen un denominador común, el software, que juega un rol crucial, como la adaptación al tamaño y recursos limitados del dispositivo. Ahora bien, la introducción de las tecnología 3G y 4G producen un incremento en la comercialización de las TIC móviles. Esto conduce a un rápida adopción de servicios móviles en combinación con aplicaciones móviles comerciales. A pesar de esta gran oportunidad de negocio, muy pocas publicaciones científicas se encuentran actualmente en la literatura para enfocar las características que deben estar presentes en un proceso de desarrollo de software en el dominio de la Computación Móvil [7], en cuyo ámbito se centra nuestro trabajo. En [22] se estudian métodos ágiles, sin embargo estos son utilizados para desarrollo de sistemas aislados y no se adaptan a un contexto de producción industrial de software que trata de familias de sistemas similares, que es nuestro caso. En vista del crecimiento de las aplicaciones para dispositivos móviles desde 2008 con el nacimiento del iPhone, Wasserman [8] realiza una encuesta para determinar las prácticas utilizadas respecto al desarrollo de software móvil, encontrando que:

1. Las aplicaciones son pequeñas, pocos miles de líneas de código en promedio, e involucran solo uno o dos responsables del desarrollo.
2. Existe una gran diferencia entre *Aplicaciones nativas* o *"Apps"* que se bajan y corren completamente en el dispositivo y *Aplicaciones "Web responsive" o "Web Apps"*, que se simplemente se despliegan en el dispositivos via un Browser, pero se ejecutan en un servidor remoto.
3. No se utiliza ningún método de desarrollo formal, ni semi-formal.
4. Los desarrollo no son documentados y pocas métricas son recopiladas.

Debe notarse que los puntos 3 y 4 son contrarios a las mejores prácticas de la IS. Sin embargo hay muchas herramientas disponibles para plataformas móviles comerciales para el desarrollo de este tipo de aplicaciones, muchas orientadas al diseño y desarrollo de la UI, cuya usabilidad es crucial para la adopción o rechazo inmediato de estas aplicaciones. Estas herramientas facilitan un desarrollo rápido y, como veremos más adelante entre los resultados de nuestro estudio (ver Sección IV.1 – Aplicación del proceso IDCM, 1.2.7), permiten garantizar el cumplimiento de muchas propiedades de calidad prioritarias requeridas por las funcionalidades de las aplicaciones. Sin embargo, como las aplicaciones móviles cobran importancia y complejidad transformándose en verdaderos sistemas de información (como por ejemplo el caso de los sistemas de salud), sería esencial aplicar buenas prácticas y sobre todos procesos sistemáticos bien definidos de IS, para asegurar el desarrollo de aplicaciones móviles de alta calidad, por lo menos en cuanto a seguridad, usabilidad y tolerancia a fallas. Wasserman [8] recomienda Scrum [23] como marco de desarrollo ágil que puede adaptarse a este dominio, para introducir mejores prácticas en el desarrollo de estas aplicaciones, sin embargo también afirma que es un reto atacar la variabilidad creciente de las TIC en este tipo de desarrollo; nuestro enfoque toca justamente este problema, considerando un modelo de variabilidad integrado a la RA. En general, la especificación de RNF muy cambiantes sigue siendo un problema abierto para manejar la evolución de estas aplicaciones en el dominio de la Computación Móvil.

II.3 Trabajos Relacionados

Se mencionó anteriormente que muy pocos trabajos científicos se han dedicado a los procesos de desarrollo de software en el dominio de la Computación Móvil y menos aún en un contexto LPS, que es el punto de interés de nuestro trabajo.

En [24] se propone un método para el desarrollo de UI especializadas para el dispositivo, en el dominio de la Computación Móvil. El método y herramienta propuesto, GeMMINi, utiliza técnicas de Desarrollo Dirigido por Modelos o *"Model Driven Engineering"* combinadas con técnicas de LPS. El método, el cual no es especificado formalmente ni semi-formalmente, permite describir de manera abstracta los requisitos de interfaz, describir las variantes de los dispositivos, transformar modelos y generar código para obtener implementaciones nativas en los dispositivos. Por último, se define un catálogo de patrones que permite representar las correspondencias entre conceptos abstractos de interfaz y especificaciones concretas de cada tipo de dispositivo. La Interacción con el usuario se especifica en "Unidades" mediante diagramas de clases UML, que complementan las descripciones de las estructuras de datos. La descripción de propiedades y variantes de dispositivos es dada mediante modelos de características [25], enfoque muy utilizado en LPS. Se basan en una ontología [26] que especifica algunos aspectos

de interacción, como la entrada por voz o ratón o la salida por proyector o pantalla y que refina los modos de entrada y salida, e incorpora otras como el manejo del dispositivo (1 ó 2 manos) o su posición (cerca o lejos del usuario). Se dispone de un catálogo de soluciones específicas para cada dispositivo a partir de conceptos abstractos de interfaz. Permite configurar la transformación con la cual se definen bocetos a partir de los cuales se generan prototipos de la UI. Se diferencian varios tipos de entidades para diferentes implementaciones en función del dispositivo. Respecto a nuestro trabajo, observamos que este proposición solo toca el componente UI, lo cual es una limitación y solo trata la variabilidad funcional y no toca el problema de la variabilidad de RNF, ni siquiera en el componente UI. Por otra parte, aunque desde el punto de vista teórico este enfoque es interesante, ya existen toolkits para adecuar la UI al dispositivo, como por ejemplo IONIC[2] y algunas plataformas Java.

En Losavio, Ordaz y Esteller [9] se presenta un proceso de diseño bottom-up extractivo, especificado semi-formalmente, repetible y sistemático, el cual es la base del presente trabajo, para construir una RA en un contexto LPS, en el dominio de los Sistemas Integrados de Salud. El *enfoque extractivo* comienza con una colección de productos existentes e incrementalmente se refactorizan sus elementos arquitecturales. En este caso se realiza un análisis de productos existentes del mercado, a partir de los cuales, mediante un análisis de similitud semántica de componentes, se reconstruyen las arquitecturas de los productos a partir de la información o documentación disponible, determinándose componentes comunes y variantes. Una contribución importante del trabajo es considerar la arquitectura especificada por un grafo conexo no dirigido, lo cual facilita realizar automáticamente la unión de los grafos de las arquitecturas de los productos, para construir una arquitectura inicial candidata (o *CA: Candidate Architecture)*, que puede ser luego completada manualmente con componentes adicionales, para así obtener la RA. Las propiedades de calidad requeridas por los componentes que representan funcionalidades básicas son consideradas desde el comienzo del proceso y pueden completar la arquitectura inicial CA, utilizando un enfoque de metas [27]. Las propiedades de calidad son especificadas por el modelo de calidad del estándar ISO/IEC 25010 [28]. Sin embargo solo se realiza la fase de análisis del dominio y no se utiliza el framework ISO/IEC 26550 [3] para la ILPS; la fase de Alcance en IDCM reduce el esfuerzo de las subsiguientes fases de diseño de la RA y además, no utiliza el enfoque de metas, el cual no ofrece una notación estándar y dificulta la visualización de componentes en caso de sistemas complejos; en cambio IDCM utiliza un enfoque basado en escenarios, mediante una tabla de trazabilidad entre componentes que representan RF y requieren propiedades de calidad (derivadas

[2] www.tutorialspoint.com/ionic/

de RNF), y los componentes que las proporcionan.

El trabajo de Koziolec y otros [29], en el dominio de la robótica, describe un proceso el cual inspiró el trabajo de Losavio, Ordaz y Esteller [9] descrito anteriormente, pero no contempla un modelo de grafo para la representación de la RA. Utiliza también una estrategia bottom-up extractiva, para reconstruir las arquitecturas de los productos existentes de una empresa y a partir de ellos construir manualmente la RA. Se utilizan técnicas de reingeniería, algunas a partir del código fuente, para reconstruir manualmente las arquitecturas de tres productos de una empresa cuya industria se basa en el uso de robots y la RA es también construida manualmente, sin pasar por una arquitectura intermedia construida automáticamente, como en [9]. La consideración de los RNF que debe satisfacer la RA es muy informal y no se utilizan estándares para especificar las propiedades de calidad relativas a los RNF.

*S*e observa que los trabajos [9] y [29] no ofrecen herramientas de soporte a los procesos planteados, aunque dicen estar previstas en trabajos futuros.

III. Proceso de Ingeniería del Dominio de la Computación Móvil (IDCM)

En IDCM, como ya fue mencionado, se consideran tres fases del ciclo ID: *1. Alcance (ALC)* con las actividades *Portafolio de Productos, Análisis del Dominio Ágil (ADA), Identificación de Activos*; *2. Ingeniería de Requisitos del Dominio* con la *Identificación de componentes comunes* y *variantes* y la *Construcción de la Arquitectura Candidata* y *3. Diseño del Dominio* con la *Construcción de la Arquitectura de Referencia* y la *Elaboración de un Documento de Recomendaciones Generales (DRG)*; el proceso completo se ilustra en las Figuras 2 y 3. En la Fase 1 ALC es donde se concentra el mayor esfuerzo de trabajo, pero esto conlleva la reducción de esfuerzo en las subsiguientes fases. Cuando nos referimos a ADA [19] no pretendemos usar ninguno de los métodos denominados "agiles" existentes en la literatura para el desarrollo de software [12] [13] [14], sino solo incorporar algunas técnicas inspiradas en los métodos ágiles, basadas en realizar entrevistas o "quiz" y reuniones con los expertos del dominio, para llegar a un consenso sobre las soluciones arquitectónicas propuestas. El número de entrevistas y reuniones debe ser determinado de acuerdo a la complejidad que se observe en la primera reunión para levantar la información sobre el dominio o sub-dominio de la empresa. Debe recordarse que el dominio de la Computación Móvil parece implicar un desarrollo ágil para aplicaciones y/o sistemas aislados [8], pero en este caso estamos en un contexto LPS de familias de sistemas, al cual no se adaptan bien los métodos ágiles. Como se mencionó anteriormente, IDCM sigue un desarrollo ascendente o

"bottom-up" extractivo inspirado en [9], respecto al estudio de productos existentes de un dominio, o cada filial o sub-dominio de ser el caso, ya desarrollados por la empresa, para construir o reconstruir sus arquitecturas y representarlas como grafos conexos no dirigidos, como vistas lógicas de componentes y conectores en el lenguaje notacional estándar UML 2.0 [5] o culaquier versión posterior que incluya descipciones arquitectónicas; para esto se extrae información de entrevistas y reuniones con los equipos de trabajo en la empresa y se diseña la arquitectura utilizada en cada aplicación, producto, o sistema identificado, cuya especificación es generalmente ausente, limitándose en algunos casos solo a diagramas de despliegues físicos de componentes. Una vez identificadas las arquitecturas y especificadas en UML, se procede a realizar automáticamente una "unión" de los grafos correspondientes para obtener los componentes *comunes* y *variantes* para el dominio o sub-dominio. Debe recordarse que una RA es una arquitectura abstracta y se caracteriza por su *modelo de variabilidad* [9] [20], a partir del cual se puede realizar la instanciación. Los componentes comunes a los productos se identifican desde el punto de vista semántico, considerando el desempeño de tareas funcionalmente similares y se procede a unificar sus nombres cuando éstos son diferentes. Los componentes variantes son los que no son comunes; estos se aglutinarán, previo otro análisis de similitud semántica para determinar el tipo de tarea desempeñada por las variantes, en los puntos de variación [20], los cuales son conjuntos de componentes variantes.

A. Fase 1. Alcance (ALC) – se repite para cada sub-dominio, si hay más de uno (ver Figura 2)

Entrada: En cada filial o sub-dominio de la empresa, se realizan entrevistas o un "quiz" y discusiones con el personal especializado, se estudia la eventual documentación existente sobre los sistemas desarrollados, toolkits y/o plataformas utilizados para el desarrollo;

Actividades:

1.1 Portafolio de Productos (PortProd) para cada sub-dominio [3]

1.1.1 Identificación de principales sistemas y/o aplicaciones desarrollados por el consorcio, sus funcionalidades (componentes y/o módulos) principales y posibles restricciones;
1.1.2 Principales sistemas, toolkits y/o plataformas de soporte utilizados para el desarrollo: funcionalidades (componentes o módulos) principales realizados

por la empresa; lenguajes, SO, herramientas, etc.;
1.1.3 Identificación de otros productos existentes del mercado, similares a los desarrollados por las filiales, sus funcionalidades (componentes y/o módulos) principales y posibles restricciones;
1.1.4 Identificación de futuros productos que podrían ser desarrollados por la empresa en general;

1.2. Análisis del Dominio Ágil (ADA) para cada sub-dominio [9] [19]

1.2.1 Estilo (s) arquitectural (es) utilizado (s);
1.2.2 Levantamiento informal de RF y RNF prioritarios;
1.2.3 Modelo de calidad *(DQM: "Domain Quality Model")*, especificado por ISO/IEC 25010 [28]);
1.2.4 Análisis de similitud semántica de componentes entre los diferentes sistemas desarrollados por la filial, representado por una *tabla de componentes y conectores (CCT)* con los principales componentes manejados por cada sistema desarrollado (nombres unificados) y como están relacionados. Se utiliza el enfoque de diseño extractivo bottom-up [9];
1.2.5 Representación en UML 2.0 [5] de las arquitecturas de los principales sistemas existentes desarrollados por la filial, con nombres de componentes unificados. Se representa la vista lógica de la arquitectura;
1.2.6 Se establece la trazabilidad entre *componentes arquitecturales "funcionales"* (percibidas directamente por el usuario) y *componentes arquitecturales "no funcionales" o funcionalidades implícitas* [28], generalmente no percibidas por el usuario y que resuelven propiedades de calidad asociadas a RNF; los componentes funcionales requieren para su buen funcionamiento o idoneidad funcional el cumplimiento de propiedades de calidad específicas, proporcionadas por las funcionalidades implícitas;

Observación 1: En el caso de la Computación Móvil, las funcionalidades implícitas suelen corresponder con funcionalidades realizadas por plataformas. toolkits o frameworks de desarrollo que ayudan a resolver las propiedades de calidad requeridas; éstas son representadas también como componentes arquitecturales. Es decir que la selección de plataformas idóneas para el desarrollo de aplicaciones Web contribuye a la calidad global de los productos generados. Las relaciones o "trazabilidad" entre estos tipos de componentes se representan en la *Tabla TraT* para cada producto de la filial o sub-dominio;

1.3 Identificación de Activos (IdAct) para cada sub-dominio

1.3.1 Determinar los principales activos o "assets" (elementos reutilizables) comunes y variantes que son manejados; la especificación de la estructura y organización del repositorio no es tratado en IDCM, el repositorio debe ser diseñado e implementado por la empresa como un nuevo proyecto. Consideramos la *Tabla IdAct* como un resultado importante de IDCM;

1.4 Construcción de la tabla global de activos comunes y variantes (IdActGlob) del consorcio

1.4.1 Renombrar la tabla IdAct como IdActGlob, a la cual se agregará el IdAct de cada sub-dominio, para obtener una tabla global de activos comunes y variantes para la empresa, al final del proceso;

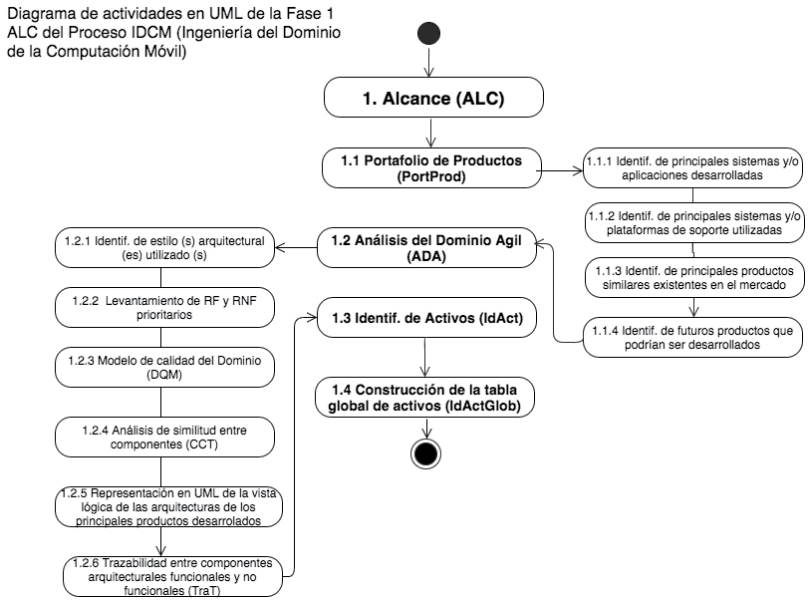

Fig. 2. IDCM: Diagrama de actividades en UML de la Fase 1 ALC. Fuente: autores

Salida: Documento ALC: texto; incluye DQM_i: tabla; IdActi: tabla; IdActGlobi: tabla; PortProdi: texto, donde *i=1,2,..n*, *n* número de sub-dominios; arquitecturas de productos del sub-dominioi: diagrama UML; CCTj: tabla, TraTj: tabla, donde *j=1,2,..p*, *p* número de productos;

Observación 2: De acuerdo con la práctica ágil [19], la primera versión entregable del ALC será sometida a discusión y eventuales modificaciones con el personal responsable, hasta llegar a un consenso. Las tablas y arquitecturas se mostrarán en detalles en la Sección 4.

B. Fase 2 Ingeniería de Requisitos del Dominio [9] – se repite para cada sub-dominio, si hay más de uno (ver Figura 3)

Entrada: ALC

Actividades:

2.1 Construcción automática de una arquitectura candidata (CA: "Candidate Architecture"), considerando productos y plataformas existentes a partir de PortProd, CCT y TraT;

2.1.1 Se realiza la "unión" de las representaciones UML (grafos) de los productos existentes;

2.2 Construcción de la representación de CA en UML 2.0;

2.3 Construcción del Modelo de Calidad Extendido (*EQM: "Extended Quality Model"*), la cual es otra forma de representar CA considerando las propiedades de calidad requeridas o funcionalidades implícitas, por cada componente funcional y posibles restricciones; se construye de las tablas TraT.

2.4 Si necesario, actualización de CA considerando componentes tecnológicos adicionales que pueden no haber sido contempladas en la unión de los grafos de las arquitecturas de los productos existentes, que satisfacen propiedades de calidad requeridas por cada componente funcional:

2.4.1 Construcción de la representación UML 2.0 de CA actualizada;

2.4.2 Actualización de EQM;

2.4.3 Actualización de IdAct;

2.4.4 Actualización de IdActGlob;

2.5 Si no, actualizar IdAct e IdActGlob respecto a la CA obtenida por la unión de los productos existentes;

Salida: CA_i: UML, EQMi: tabla, IdActi: tabla, IdActGlobi donde *i=1, 2, ... n; n* número de sub-dominios;

C. Fase 3. Diseño del Dominio [9] – se repite para cada sub-dominio, si hay más de uno (ver Figura 3)

Entrada: CA_i, EQMi, IdActi donde *i=1, 2, ... n; n* número de sub-dominios;

Actividades:

3.1 Para cada sub-dominio construcción de una Arquitectura de Referencia (RA):

3.1.1 Determinación de los puntos de variación (o *"variation points"* o

"placeholders"), los cuales agrupan componentes que desempeñan tareas similares y serán instanciados en el ciclo IA para generar un producto concreto de la familia de productos considerada en el dominio;

3.1.2 Construcción de la representación UML 2.0 de RA;

3.2 Elaboración de *DRG: Documento de Recomendaciones Generales* para el sub-dominio, que contempla análisis de los resultados, limitaciones y recomendaciones para posibles mejoras.

Salida: CA_i actualizada: UML, EQMi actualizada: IdActi actualizada; IdActGlobi actualizada; RAi: UML, DRGi: texto, donde *i=1, 2, ... n, n* número de sub-dominios;

Observación 3: De acuerdo con prácticas inspiradas en los métodos ágiles [19], las salidas de las Fases 2 y 3 serán discutidas con el personal responsable de las filiales, con eventuales modificaciones a ser realizadas, hasta llegar a un consenso. Las tablas y arquitecturas se mostrarán en detalles en la Sección 4 para un sub-dominio particular.

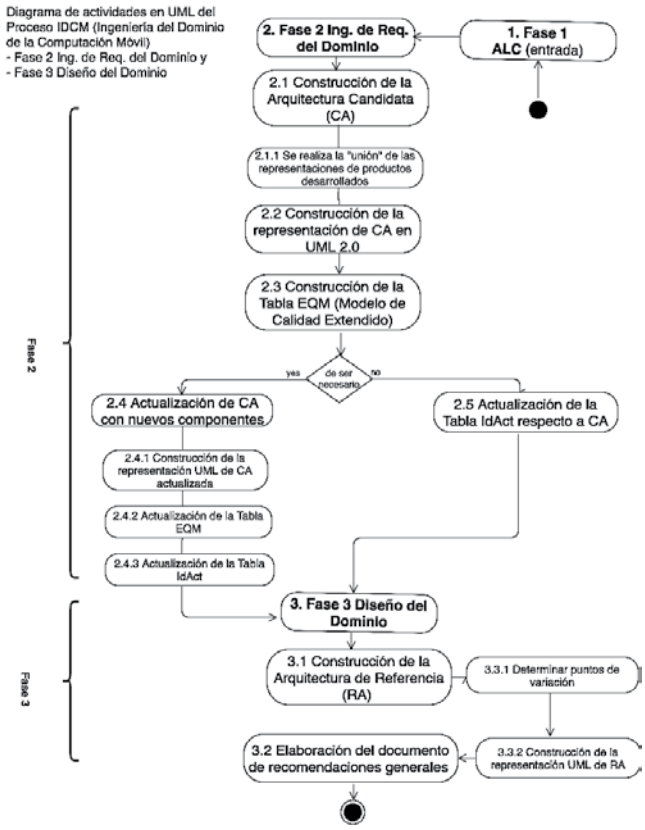

Fig. 3. IDCM: Diagrama de actividades en UML de las Fases 2. Ing. de Requisitos del Dominio y 3. Diseño del Dominio. Fuente: autores

IV. Aplicación del proceso IDCM al caso del consorcio CONECTIUM LIMITED C.A. Venezuela

El proceso *IDCM* se ha aplicado sucesivamente a las tres filiales o sub-dominios TransaMóvil, Sinapsis y Conectium del consorcio CONECTIUM Limited C.A. Venezuela, donde se utilizan TIC de rápida evolución que caracterizan el dominio de la Computación Móvil. La ID realizada mediante IDCM permite hacer reingeniería sobre la descripción y documentación existente de productos de software realizados en la empresa, sin bajar a nivel del código fuente.

Un primer proyecto trazado para este consorcio fue determinar *Componentes Reutilizables en el dominio de la Computación Móvil (CRCM)*, porque al comenzar aun no estaba claro si se podía llegar a una arquitectura de referencia para cada sub-dominio y menos aún para todo el consorcio. CRCM consiste en la identificación de activos reutilizables, es decir componentes de software comunes y variantes que son utilizados en los desarrollos de sistemas o aplicaciones de software realizados por la empresa; sin embargo para identificar los eventuales activos, hubo que realizar un proceso completo de reingeniería de alto nivel para diseñar las arquitecturas de cada producto elaborado en cada sub-dominio, que no habían sido consideradas por la empresa, por lo tanto se definió el proceso IDCM el cual incluye CRCM como una de las actividades, porque en paralelo se construye el repositorio de activos.

El consorcio CONECTIUM desarrolla software móvil, basado en TIC multimedia dirigido a laptop, tableta, celular o cualquier otro dispositivo móvil con conexión inalámbrica. El consorcio comprende tres sub-dominios o filiales:

1. *Contenidos de entretenimiento (Conectium, desarrollo inicial del consorcio)*
2. *Transacciones financieras (TransaM*óvil)
3. *Sistemas médicos (Sinapsis, desarrollo más reciente)*

Los tres sub-dominios fueron estudiados en el siguiente orden: 1. TransaMóvil, 2. Sinapsis y 3. Conectium. El tiempo total de realización del estudio fue de seis meses, desde julio a diciembre 2016. Se identificaron como activos más de 70 componentes. IDCM será aplicado paso a paso al sub-dominio TransaMóvi, para ilustrar el proceso completo.

IV.1 Aplicación de IDCM al sub-dominio TransaMóvil

A. Fase 1 Alcance (ALC) – sub-dominio TransaMóvil – Figura 2

Entrada: Un quiz o cuestionario y tres reuniones de discusión con el personal especializado para el estudio de la documentación existente sobre los sistemas desarrollados y/o utilizados y de productos existentes en el mercado; a partir del

quiz y de las reuniones se obtuvo la siguiente información sobre el negocio:

Objetivo del negocio: operaciones de carácter financiero en celulares, usando tecnología web.

Reglas del negocio:

- garantizar la protección de usuarios y sus transacciones financieras: implica satisfacer la propiedades de calidad *seguridad (autenticidad, confidencialidad, integridad);* esencialmente se garantiza el control de acceso [28]; las exigencia de estándares de seguridad de datos financieros tipo *PCI DSS (Payment Card Industry Data Security Standard)* por ahora son manejados por terceros.

- garantizar la confiabilidad de las transacciones financieras, implica satisfacer la propiedad de calidad *confiabilidad-robustez*, es decir la habilidad de un sistema de tratar errores en ejecución y entrada de datos [28] es manejada por la plataforma transaccional propietaria Kinaku[3]; los mecanismos responsables de satisfacer la capacidad de mantenimiento o mantenibilidad respecto a la UI es la plataforma AngularJs[4].

- usar el servicio de nube propietario AWS[5] (Amazon Web Services)

Actividades:

1.1 Portafolio de Productos (PortProd) para TransaMóvil [3]

1.1.1 Identificación de principales sistemas y/o aplicaciones desarrollados: funcionalidades (componentes) principales, posibles RNF como restricciones o propiedades de calidad exigidas y plataformas utilizadas por cada componente; esta información se recopiló en tres reuniones con el equipo de trabajo y comunicación por e-mail:

- *Página Web Nacional – Portal via Browser - PWN*: Generar interacción directa entre el usuario y la plataforma de servicios que TransaMóvil ofrece en Venezuela.
- *Página Web Internacional (Neeru) – Portal via Browser - PWINT*: Generar interacción directa entre el usuario y la plataforma de servicios que TransaMóvil ofrece internacionalmente.
- ***Producto PWN/INT** Requisitos funcionales (RF) o principales funcionalidades ofrecidas por ambos portales:*
  - Pago de servicios
  - Ingreso saldo
  - Transferencia dinero
  - Cobranza

[3] www.ikaruscorp.com/ikarus-english/modules.html
[4] https://angularjs.org/
[5] https://aws.amazon.com/

- Afiliación de medios de pago
- Afiliación de servicios
- Notificaciones
- Gestión usuario/Perfil

Los RF son similares para ambos productos PWN y PWINT por eso se ha colocado un solo producto PWN/INT; son ubicados en el *Core de Servicios* como servicios de la Capa de Servicios de Nube AWS.

- PWN/INT - RNF del FrontEnd – Capa Presentación (ver Tabla 1):

Tabla 1. RNF de PWN/INT para FrontEnd Web/Móvil

PWN/INT FrontEnd Web/ Móvil - RNF	*Mecanismo o dispositivo responsable de satisfacer el RNF*
usabilidad	CSS3[6] para estilos en la página: depende del diseño de la página
portabilidad	AngularJs- Javascript,[7] HTML5[8], IONIC[9] SDK traductor a diferentes SO en móviles (iOS, Android)
mantenibilidad (modificabilidad)	AngularJs, con manejo del patrón de diseño MVC[10] para separar capas de Presentación/Proceso
uso de recursos (eficiencia en tiempo)	USSD[11] protocolo de comunicación similar al servicio SMS, los mensajes USSD crean una conexión en tiempo real durante una sesión USSD que permanece abierta, para un intercambio bidireccional de una secuencia de datos. Esto hace al USSD más eficiente que los servicios que utilizan SMS. Bootstrap[12] para desarrollo rápido del FrontEnd

Los RNF son también similares para ambos productos PWN y PWINT, por esto se presentan en el producto PWN/INT (ver Tabla 2).

PWN/INT - RNF del BackEnd – Capa Proceso (ver Tabla 2):

6 Cascading Style Sheets
7 https://www.javascript.com/
8 HyperText Markup Language, versión 5
9 www.tutorialspoint.com/ionic/
10 Model View Controller
11 Unstructured Supplementary Service Data
12 getbootstrap.com/

Tabla 2. RNF de PWN/INT para BackEnd

PWN/INT BackEnd - RNF	*Mecanismo o dispositivo responsable de satisfacer el RNF*
seguridad (autenticidad, confidencialidad, integridad)	Protocolos de comunicación HTTPS para control de acceso; estándares de seguridad financieros tipo PCI DSS, a ser desarrollados por terceros
portabilidad	NodeJs[13] - Javascript (capa de proceso), ORM[14] para pasar de BD relacional a objetos en Java (Capa de Datos)
confiabilidad (robustez)	NodeJs-Express-SailsJs[15] para mejorar NodeJs
confiabilidad (disponibilidad-persistencia)	PostgreSQL[16] como sistema manejados de BD

- ***Producto Monedero Electrónico*** *– acceso por Portal en Browser:* permitir a los usuarios de la plataforma de servicios de TransaMóvil o a terceros, poder ingresar, utilizar y extraer saldo de diferentes medios de pago. Es utilizado por PWN/INT y GasMóvil.
- Monedero Electrónico RF:
  - Validación de ingreso de saldo por deposito o transferencia
  - Cierre de solicitudes de ingreso de saldo
  - Monitoreo de la cuenta monedero
  - Bloqueo de cuentas.
  - Administración de cuentas
  - Monedero Electrónico RNF del FrontEnd – Capa Presentación (ver Tabla 3):

Tabla 3. RNF de Monedero Electrónico para FrontEnd Web

Monedero Electrónico FrontEnd Web - RNF	*Responsable de satisfacer el RNF*
usabilidad	CSS3 para estilos en la página: depende del diseño de la página
portabilidad	AngularJs-Javascript, HTML5, IONIC
mantenibilidad (modificabilidad)	AngularJs, con manejo de MVC para separar capas Presentación/Proceso
uso de recursos (eficiencia en tiempo)	USSD, Bootstrap

[13] https://es.wikipedia.org/wiki/Node.js
[14] Object Relational Mapping
[15] runastartup.com/express-js-vs-sails-js-comparison/
[16] https://www.postgresql.org/

Monedero Electrónico no tiene FrontEnd Móvil.

- *Monedero Electrónico RNF del BackEnd - Capa Proceso (ver Tabla 4):*

Tabla 4. RNF de Monedero Electrónico para BackEnd

Monedero Electrónico BackEnd - RNF	*Mecanismo o dispositivo responsable de satisfacer el RNF*
seguridad (autenticidad, confidencialidad, integridad)	HTTPS, estándares de seguridad financieros tipo PCI DSS a ser desarrollados por terceros
portabilidad	NodeJs-Javascript (capa de proceso), ORM para pasar de BD relacional a objetos Java (Capa de Datos)
confiabilidad (robustez)	NodeJs-Express-SailsJs
confiabilidad (disponibilidad-persistencia)	PostgreSQL

Los RNF de este producto y las plataformas que ayudan a resolverlos son básicamente iguales a los del BackEnd de PWN/INT; como Monedero Electrónico no tiene FrontEnd móvil, IONIC no está presente.

- ***Producto GasMovil:*** Permitir la compra de combustible en móviles; el cobro se realiza mediante la plataforma de servicios de TransaMóvil AWS
- GasMóvil RF:
  - Gestión Usuario/Perfil - solo autenticación
  - Pagar combustible
  - Gestión de transacciones
  - Determinar ubicación por geolocalización
- GasMóvil RNF - Capas Presentación y Proceso:

Igual que para PWN/INT, en FrontEnd se utiliza también IONIC pero en el dispositivo se traduce solo para el SO Android.

- ***Producto Conector Pagos de Servicios*** *(Servicio Web):* Realizar la comunicación con los sistemas de proveedores de servicios o productos y para registrar el pago de los mismos.
  Es utilizado por PWN/INT y GasMóvil.
- Conector Pagos RF:
  - Pagar servicios (Movistar, Digitel, Inter).
  - Consultar transacciones
  - Consultar servicios
  - Consultar saldo pendiente (servicios post pago)

  Conector Pagos no tiene FrontEnd; es un servicio Web utilizado por los productos PWN/INT y GasMóvil.
- *Conector Pagos RNF del BackEnd – Capa Proceso (ver Tabla 5):*

Tabla 5. RNF de Conector Pagos para BackEnd

Conector Pagos BackEnd - RNF	*Mecanismo o dispositivo responsable de satisfacer el RNF*
seguridad (autenticidad)	Protocolos de comunicación HTTPS, estándares de seguridad financieros tipo PCI DSS a ser desarrollados por terceros
portabilidad	Spring para Java (Capa de Proceso), ORM para pasar de BD relacional a objetos en Java (Capa de Datos)
confiabilidad (robustez) eficiencia	Kinacu
Confiabilidad (disponibilidad persistencia)	Kinacu con SQL Server, en la Capa de Servicios de Nube AWS
Corrección-Precisión	Kinacu en la Capa de Servicios de Nube AWS

1.1.2 Identificación de productos existentes del mercado (fuente Quiz y reuniones), similares a los desarrollados por las filiales, sus funcionalidades (componentes) principales y posibles restricciones:
paypal.com, xoom.com, ding.com, mispagosprovincial.com, instapago.com, simple.com, mercadopago.com
1.1.3 Identificación de futuros productos que podrían ser desarrollados por TransaMóvil en general (fuente quiz):
- Ampliar Monedero Electrónico:
- Pasarela de pagos para recibir peticiones de cobro y concentrar todos los pagos con tarjeta de crédito
- Ampliar la capacidad de pago a personas jurídicas; distribuir saldo, otorgar créditos, etc.
- Estudio de core bancarios open source para la gestión de transacciones bancarias como Open Bank y FinTP (Finkers United), para ser integrados a Kinacu (popietario); satisfacen estándares ISO 8583 para manejo de transacciones bancarias.

El Portafolio de Productos del sub-dominio TransaMóvil (PortProd) está constituido por:

Producto PWN/INT – Portal via Browser, con FrontEnd Web/Móvil

Producto GasMóvil – Portal via Browser, con FrontEnd Web/Móvil

Observación 1. A efectos de construir la arquitectura candidata CA en la fase 2 del proceso IDCM mediante la unión de las arquitecturas de los productos desarrollados por TransaMóvil, solo consideramos PWN/INT y GasMóvil como productos porque Monedero Electrónico y Conector Pagos son utilizados en ambos productos como componentes.

1.2. Análisis del Dominio Ágil (ADA) para TransaMóvil [19]

1.2.1 Estilo (s) arquitectural (es) utilizado (s):
estilo híbrido o combinación de estilos, basado en eventos, capas (Presentación, Proceso y Datos) con comunicación bajo modelo cliente-servidor por Internet via HTTPS; el protocolo USSD (Unstrucured Supplementary Service Data) es utilizado en la Capa de Presentación para incrementar eficiencia en tiempo; UI en cliente (Browser) separada de la Capa de Proceso por el uso del patrón arquitectural MVC por AngularJs; Capa de Servicios de Nube para almacenamiento e infraestructura - IaaS (Infrastructure as a Service) AWS, donde se ha instalado la plataforma Kinacu para garantizar eficiencia, confiabilidad y corrección. AWS y Kinacu en general implican satisfacer eficiencia en cuanto al uso de recursos-escalabilidad, seguridad por mantener datos centralizados, mantenibilidad (modificabilidad, reutilización), interoperabilidad de servicios y disponibilidad-persistencia de datos. El esquema general de la arquitectura en notación informal se presenta en la Figura 4.
1.2.2 Levantamiento informal de requisitos funcionales (RF) y no funcionales (RNF)
prioritarios para el sub-dominio de TransaMóvil:

RNF prioritarios:

1. seguridad (autentificación, confidencialidad, integridad),
2. portabilidad,
2. eficiencia,
3. mantenibilidad,
2. usabilidad (en el diseño de portales del frontend),
3. corrección-precisión

donde $1 \leq prioridad \leq 3$, 1 indica máxima prioridad; las propiedades de calidad son definidas de acuerdo al estándar ISO/IEC 25010 [28]. Estos RNF se han obtenido estudiando los RNF asociados a cado producto y los RNF relativos al estilo arquitectónicos para este sub-dominio.

RF prioritarios:

PWN/INT (Pagina web Nacional- Página web Internacional (Neeru), Monedero Electrónico (administración monedero), GasMóvil. El servicio Conector de Pagos es utilizados por las funcionalidades prioritarias PWN/INT y GasMóvil

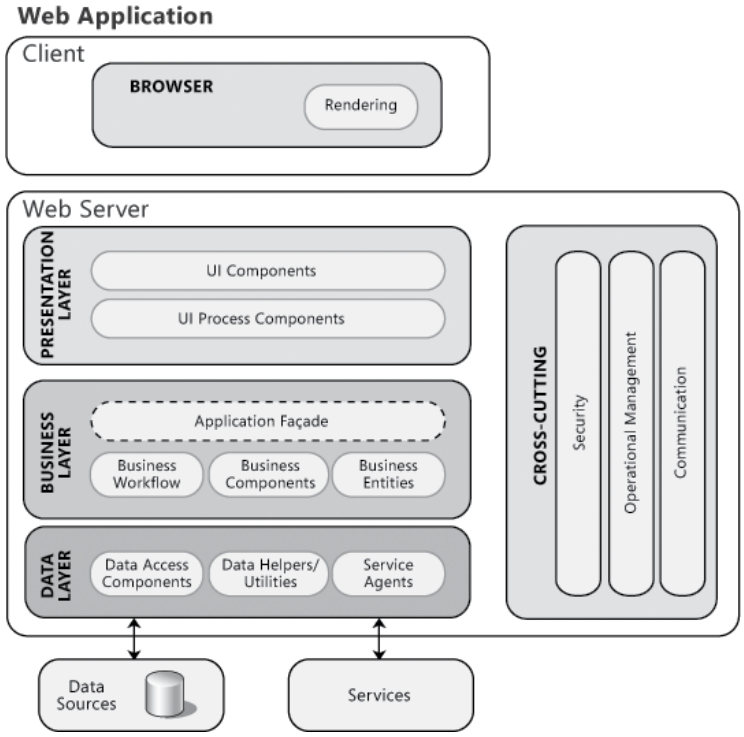

Fig. 4. Arquitectura de estilo híbrido Basado en Eventos/Capas, comunicación modelo Cliente-Servidor. Fuente: Wikipedia

1.2.3 Objetivos y/o reglas del negocio y correspondencias con RF y RNF
Se definieron como entrada a la Fase 1 de Alcance, a partir del Quiz y las reuniones realizadas con el personal del sub-dominio TransaMóvil.
1.2.4 Modelo de calidad (DQM, especificado por ISO/IEC 25010 [28] características y sub-características:
seguridad (autentificación, confidencialidad, integridad), portabilidad, mantenibilidad (modificabilidad, reutilización), usabilidad (frontend), eficiencia en tiempo y en recursos, confiabilidad (robustez, disponibilidad), idoneidad funcional (corrección-precisión), compatibilidad (interoperabilidad).

Observación 2: de las tablas para describir los RNF de cada producto de TransaMóvil presentadas en la actividad 1.1.1, se observa que las plataformas, toolkits o mecanismos utilizados para el desarrollo del software, permiten en gran parte satisfacer las propiedades de calidad exigidas por los RNF de cada producto. Por lo tanto se formulan las siguientes premisas importantes, como resultado del

estudio realizado para TransaMóvil:

Premisas

- Las plataformas de desarrollo de las aplicaciones o productos de TransaMóvil son en gran parte las responsables de satisfacer los requisitos de calidad (RNF) relativos a las funcionalidades principales (RF)
- Es recomendable la selección de plataformas adecuadas para responder a los RNF prioritarios al implementar los RF.

1.2.5 Análisis de similitud semántica de componentes entre los diferentes sistemas desarrollados por la filial, representado por una Tabla de Componentes y Conectores (CCT) con los principales componentes manejados por cada sistema desarrollado (nombres unificados) y como están relacionados. Se utiliza un enfoque extractivo "bottom-up" [9]; los componentes comunes (CC) se resaltan en gris; *"R"* indica la relación o conector entre dos componentes.
1.2.6 Representación en UML 2.0 de las arquitecturas de los principales sistemas existentes desarrollados por TransaMóvil, con nombres de componentes unificados. Se representa la vista lógica [4] de la arquitectura (ver Figura 5).
1.2.7 Por lo expresado en el punto 1.2.4, se establece la trazabilidad entre *componentes arquitecturales "funcionales"* (percibidas directamente por el usuario) y *componentes arquitecturales "no funcionales" o funcionalidades implícitas*, generalmente no percibidas por el usuario y que resuelven propiedades de calidad que corresponden a los RNF. Los componentes funcionales, que representan funcionalidades básicas correspondientes a los RF, requieren para su buen funcionamiento o idoneidad funcional el cumplimiento de propiedades de calidad específicas, resueltas por estas funcionalidades implícitas. En el caso de la computación móvil, analizada aquí en el desarrollo realizado por TransaMóvil, las funcionalidades implícitas corresponden al uso de plataformas o frameworks de desarrollo que ayudan a resolverlas y que se han representado también como componentes arquitecturales.
Es decir que la selección de plataforma idóneas de desarrollo de aplicaciones Web contribuye a la calidad global de los productos generados. Las relaciones o "trazabilidad" entre estos componentes se representan en las tablas TraT para cada producto, ver Tablas 10 y 11.

1.3 Identificación de Activos (IdAct) para cada sub-dominio

1.3.1 Determinar los principales activos o "assets" (elementos reutilizables) comunes y variables que son manejados y que formarán parte del repositorio de cada sub-dominio; se deducen de las tablas CCT (ver Tablas 6, 7, 8 donde

los comunes son resaltados en gris) y se visualizan también en las Figuras 4 y 5 de las arquitecturas. El repositorio de activos deberá ser diseñado e implementado por la empresa como un nuevo proyecto, de ser el caso; en este trabajo nos limitamos a identificar los activos que se dedujeron de los productos desarrollados en la empresa para cada sub-dominio. Consideramos IdAct. (ver Tabla 9), como un resultado importante de este proceso. Se realiza en paralelo con las actividades 1 y 2.

Nótese que los componentes *b17. Monedero Electrónico* y *b15. Conector de Pagos* que es un servicio Web, son especificados como sub-componentes del componente *b2. BackEnd* en PWN/INT y en GasMóvil.

Tabla 6. CCT – Tabla de Componentes y Conectores de cada producto - Capa Presentación

Producto PWN/INT	Producto GasMóvil
a1. FrontEnd	a1
a2. Portal Web	a2
a3. Portal Móvil	a3
a4. Browser	a4
a5. AngularJs	a5
a6. CSS3	a6
a7. Bootstrap	a7
a8. HTML5	a8
a9. IONIC	a9
a10. USSD	-
Conectores	
a1Ra2	a1Ra2
a1Ra3	a1Ra3
a2Ra4	a2Ra4
a2Ra5	a2Ra5
a2Ra6	a2Ra6
a2Ra7	a2Ra7
a2Ra8	a2Ra8
a2Rb1	a2Rb1
a3Ra4	a3Ra4
a3Ra5	a3Ra5
a3Ra6	a3Ra6
a3Ra7	a3Ra7
a3Ra8	a3Ra8
a3Ra9	a3Ra9
a3Rb1	a3Rb1
a10Rb3	-

Tabla 7. Tabla CCT – Componentes y Conectores de cada producto – Capa Proceso

PWN/INT	GasMóvil
b1. AWS	b1
b2. BackEnd	b2
b3. Servidor Open VPN/Linux (façade)	b3
b4. Core de Servicios	b4
b5. Pago de Servicios	b5
b6. Ingreso saldo	b6
b7. Trasferencia de dinero	b7
b8. Cobranza	b8
b9. Afiliación Medios de Pagos	b9
b10. Afiliación de Servicios	b10
b11. Notificaciones	b11
b12. Gestión Usuario/Perfil	b12
b13. Kinaku	b13
b14. SQLServer	b14
b15. Conector Pagos	b15
b16. Spring	b16
b17. Monedero Electrónico	b17
b18. NodeJs	b18
b19. Express	b19
b20. SailsJs	b20
-	b21. Pagar combustible
-	b22. Gestión de Transacciones
-	b23. Geolocalización
b24. Validación Ingreso Saldo Dep/Transf	b24
b25. Cierre Soicitud Ingreso Saldo	b25
b26. Monitoreo Cuenta Monedero	b26
b27. Bloqueo	b27
b28. Administración de Cuentas	b28
-	b29. RaspeberryPI
Conectores	
b1Rb3	b1Rb3
b1Rb4	b1Rb4
b1Rb13	b1Rb13
b2Rb15	b2Rb15
b2Rb17	b2Rb17
b3Rb15	b3Rb15
b4Rb5	b4Rb5
b4Rb6	b4Rb6
b4Rb7	b4Rb7
b4Rb8	b4Rb8
b4Rb9	b4Rb9
b4Rb10	b4Rb10
b4Rb11	b4Rb11
b4Rb12	b4Rb12
b13Rb14	b13Rb14

Continúa...

Conectores	
b15Rb16	b15Rb16
b15Rb5	b15Rb5
b15Rb9	b15Rb9
b15Rb10	b15Rb10
b15Rb13	b15Rb13
b17Rc1	b17Rc1
b17Rb6	b17Rb6
b17Rb7	b17Rb7
b17Rb24	b17Rb24
b17Rb25	b17Rb25
b17Rb26	b17Rb26
b17Rn27	b17Rn27
b17Rb28	b17Rb28
b17Rb18	b17Rb18
b18Rb19	b18Rb19
b19Rb20	b19Rb20
-	b21Rb17
-	b22Rb15
-	b23Rb2
-	b29Rb4

Tabla 8. CCT – Tabla de Componentes y Conectores de cada producto- Capa Datos

PWN/INT	GasMóvil
c. Capa de Datos (Servidor de Datos)	
c1. Data Base c2. Postgres c3. ORM	c1 c2 c3
Conectores	
c1Rc2 c2Rc3 c1Rb13	c1Rc2 c1Rc3 c1Rb13

1.4 Construcción de la tabla global de activos comunes y variantes (IdActGlob) para el consorcio

1.4.1 IdActGlob = $IdAct_1$, para TransaMóvil. Se completará esta actividad cuando se tengan las IdAct de los demás sub-dominios: IdActGlob = IdActGlob + $IdAct_2$ + $IdAct_3$ (ver Tabla 12);
La Tabla IdAct muestra los activos comunes a ambos productos; en este caso b21. Pagar Combustible, b22. Gestión de Transacciones y b29. RaspberryPI son componentes solo presentes en GasMóvil; son resaltados en gris en la Figura 5.

Salida para el sub-dominio TransaMóvil – Fase 1: Documento ALC: texto; el documento incluye: $DQM_{Transamóvil}$: tabla, $IdAct_{TransaMóvil}$: Tabla 9; documento $PortProd_{TransaMóvil}$: texto; CCTi: Tablas 6, 7 y 8, arquitecturas de productosi: Figura 5; TraTi: Tablas 10 y 11, IdActGlobi:Tabla 12, donde *i=PWN/INT, GasMóvil*;

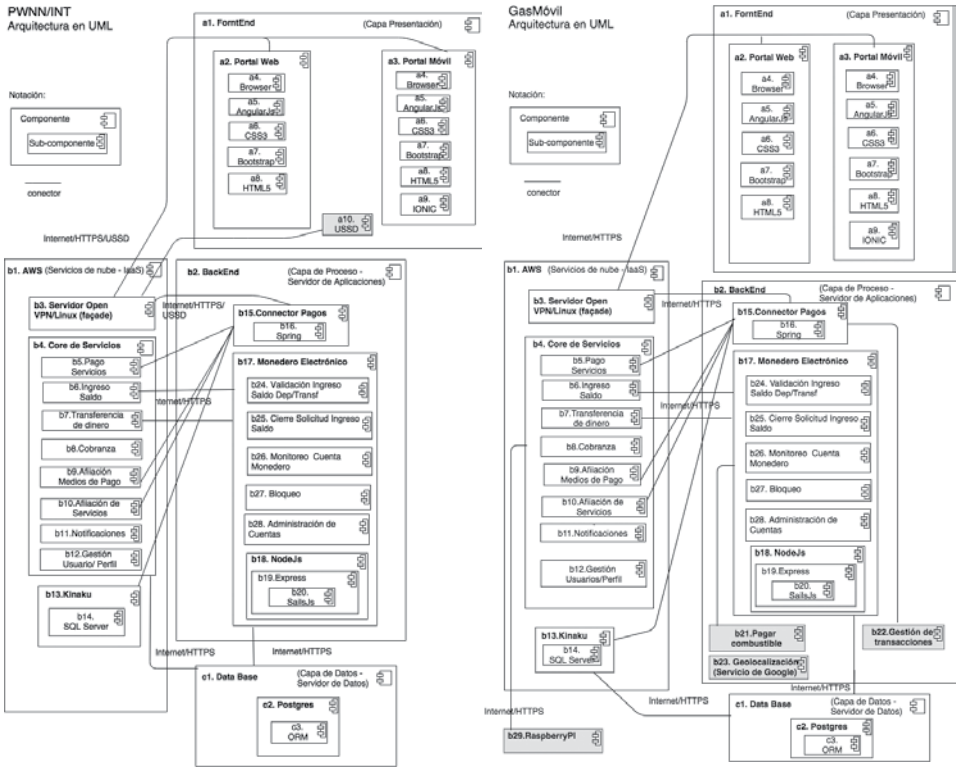

Fig. 5. Vista lógica UML de las arquitecturas de los productos PWN/INT y GasMóvil; los componentes no comunes (variantes) son resaltados en gris. Fuente: autores.

Tabla 9. IdAct: tabla de Identificación de activos para el sub-dominio TransaMóvil

Componentes de los productos PWN/INT y GasMóvil	*Comentarios*
Capa Presentación (FrontEnd)	
a1. Frontend a2. Portal Web **a3. Portal Móvil**	-a1, a2, a3 desarrollados por TransaMóvil. comunes
- a4. Browser - a5. AngularJs - a6. CSS3 - a7. Bootstrap - a8. HTML5 - a9. IONIC	- a4, a5, a6, a7, a8, a9 plataformas utilizadas para el desarrollo. comunes - a9 es común a ambos productos para a3. Portal Móvil
- a10. USSD	- a10 es variante, solo para PWN/INT
Capa Proceso (BackEnd)	
b1. AWS (servicios de nube)	- b1 Amazon Web Server, común
b2. Backend	- b2 Configurado y/o desarrollado por TransaMóvil. Común
b3. Servidor Open VPN/Linux (façade)	- b3 común
b4. Core de Servicios	- b4 contiene servicios comunes configurados y/o desarrollados por TransaMóvil
b5. Pago de Servicios - b6. Ingreso saldo - b7. Trasferencia de dinero - b8. Cobranza - b9. Afiliación Medios de Pagos - b10. Afiliación de Servicios - b11. Notificaciones - b12. Gestión Usuario/Perfil	- b5, b6, b7, b8, b9. b10, b11, b12 comunes
-b15. Conector Pagos b17. Monedero Electrónico - b16. Spring - b18. NodeJs - b19. Express - b20. SailsJs	b15, b17 desarrollados por TransaMóvil, comunes b13, b16, b18, b19, b20 plataformas utilizadas para el desarrollo, comunes

Continúa...

b21. Pagar Combustible - b22. Gestión de Transacciones - b23. Geolocalización - b24. Validación Ingreso Saldo Dep/Trans - b25. Cierre Solicitud Ingreso Saldo - b26. Monitoreo Cuenta Monedero - b27. Bloqueo - b28. Administración de Cuentas - b29. RaspberryPI	- b21, b22, b23, b29 variantes, son funcionalidades específicas, solo para GasMóvil - b24, b25, b26, b27, b28, b29 comunes
-- b13. Kinaku - b14. SQL Server	- b13 plataforma transaccional que incluye b14 como base de datos, comunes
- Protocolos HTTPS	- Protocolos de comunicación de redes, comunes
Capa Datos	
c1. Data Base	
- c2. PostgreSQL - c3. ORM	- c2 BD relacional con portabilidad a Java mediante c3 ORM, comunes

Tabla 10. TraT PWN/INT: Tabla de Trazabilidad entre los componentes del producto PWN/INT, propiedades de calidad requeridas y plataformas o mecanismos que las resuelven

Componentes del producto PWN/INT	*Propiedades de calidad requeridas por el componente*	*Propiedades de calidad proporcionadas por la plataforma*	*Comentarios*
Capa Presentación (FrontEnd) - a1. FrontEnd - a2 Portal Web - a3 Portal Móvil	- usabilidad - modificabilidad - portabilidad - eficiencia - disponibilidad	- a6. CSS3 - a5. AngularJs - a9. IONIC - a8. HTML5 - a7. Bootstrap - a10. USSD - a4. Browser	- estilo en cascada para doc. HTML - framework abierto de JavaScript-MVC - traducción a diferentes SO de dispositivos móviles (solo para a3) - lenguaje de marcas versión 5 - framework abierto soporta HTML5 y CSS3 - servicio (protocolo) de comunicación - la disponibilidad depende de la disp. del Browser

Continúa...

Capa Proceso (BackEnd) b1 AWS	- portabilidad - interoperabilidad - mantenibilidad (modificabilidad) - seguridad (autentificación, confiabilidad, integridad)	- Proveedor de Servicio de Nube - Protocolos HTTPS	- IaaS; cumplen con las propiedades mencionadas por ser servicio de nube - protocolos de comunicación de red
- b3. Servidor Open VPN/ Linux	- portabilidad - interoperabilidad - seguridad (autentificación, confiabilidad, integridad)	- es una capa façade - Protocolos HTTPS	- intermediario para conectar a2 y a3 Portales en a1. ForntEnd con b15. Conector Pagos en b2. BackEnd - protocolos de comunicación de red
- b4. Core de Servicios - b5. Pago de Servicio - b6. Ingreso Saldo - b7.	- portabilidad - interoperabilidad - seguridad (autentificación, confiabilidad, integridad) - mantenibilidad (modificabilidad)	- igual que b1. AWS por ser servicios Web	- por ser servicios Web

Continúa...

Transferencia de Dinero - Cobranza - b9. Afiliación Medios de Pago - b10. Afiliación de Servicios - b11. Notificaciones - b12. Gestión Usuario/ Perfil b15. Conector Pagos b17. Monedero Electrónico	- corrección-precisión - disponibilidad-persistencia - seguridad (autentificación) - interoperabilidad - mantenibilidad (modificabilidad) - portabilidad - confiabilidad (disponibilidad-persistencia) - eficiencia - modificabilidad - portabilidad - eficiencia - confiabilidad (robustez) - modificabilidad - seguridad - corrección-precisión - confiabilidad (disponibilidad-persistencia)	- b13. Kinacu - b14 SQL Server - protocolos HTTPS - servicio Web - b16. Spring b13. Kinacu - b18. NodeJs - b19. Express - b20. SailsJs - protocolo HTTPS - b1. Kinacu - b14. SQL Server - c2. Postgres	- plataforma para la gestión de transacciones – encriptación y autenticación de cada transacción - SMBD relacional de Microsoft; soporta c2. Postgres - por ser servicio Web - framework abierto para el desarrollo de aplicaciones Web y gestión de transacciones para Java; similar a un EJB - plataforma transaccional - framework abierto para desarrollo bajo Java; contiene Express quien usa SailsJs; diseño robusto, maneja MVC; es un ambiente para JavaScript basado en V8 motor de código abierto para JavaScript, para mejorar eficiencia - SMBD relacional - SMBD relacional con facilidades de objetos Java mediante c3. ORM
Capa Datos c1. Data Base - c2. Postgres - c3. ORM	- capacidad-escalabilidad - disponibilidad-persistencia - seguridad (integridad) - portabilidad	- SMBD - SMBD relacional - c3. ORM	- las propiedades de calidad son resultas por mecanismos propios de las bases de datos - para manejar objetos en Java

Tabla 11. TraT GasMóvl: Tabla de Trazabilidad entre los componentes del producto GasMóvil, propiedades de calidad requeridas y plataformas o mecanismos que las resuelven

Componentes del producto GasMóvil	*Propiedades de calidad requeridas por el componente*	*Propiedades de calidad proporcionadas por la plataforma*	*Comentarios*
Capa Presentación (FrontEnd) - a1. FrontEnd - a2 Portal Web - a3 Portal Móvil	- Igual que PWN/INT	- Igual que PWN/INT	- Igual que PWN/INT - Igual que PWN/INT
Capa Proceso (BackEnd) b1, b3, b4, b13, b14, b15,	- Igual que PWN/INT	- Igual que PWN/INT	- funcionalidades específicas, se mantienen las mismas propiedades de calidad de PWN/INT
b 16, b17, b18, b19	- Igual que PWN/INT	- Igual que PWN/INT	- Servicio de Google para ubicar dispositivo móvil
b2. BackEnd - b21. Pagar Combustible - b22. Gestión de Transacciones			- Comunica directamente con el chip del surtidor de combustible y el core de servicios
- b23. Geolocalización	- eficiencia, seguridad	- servicio Web	
- b29. RaspberryPI	- eficiencia, precisión, seguridad	- componente externo	
Capa Datos c1. Data Base - c2. Postgres - c3. ORM	- capacidad-escalabilidad - disponibilidad-persistencia - integridad - portabilidad	- SMBD - SMBD relacional - c3. ORM	- las propiedades de calidad son resultas por mecanismos propios de las bases de datos - para manejar objetos en Java

B. Fase 2. Ingeniería de Requisitos del Dominio (IngReq) – sub-dominio TransaMóvil - Figura 3

Entrada: Resultados de la Fase 1 ALC: documento ALC para TransaMóvil, incluye: $DQM_{Transamóvil:}$ tabla, $IdAct_{TransaMóvil:}$ Tabla 9; documento $PortProd_{TransaMóvil:}$ texto; CCTi: Tablas 6, 7 y 8, arquitecturas de productosi: Figura 5; TraTi: Tablas 10 y 11, IdActGlobi:Tabla 12, donde *i=PWN/INT, GasMóvil*;

Actividades:

2.1 Construcción de una arquitectura Candidata (CA) utilizando un enfoque extractivo "bottom-up", considerando productos y plataformas existentes a partir de PortProd, CCT y TraT;

Recordemos que PortProd para TransaMóvil está constituido por los productos PWN/INT y GasMóvil;

2.1.1 Se realiza automáticamente la unión de los grafos de las arquitecturas de los productos de TransaMóvil (ver en la Figuras 4 y 5 sus representaciones en UML).

2.2 Representación en UML 2.0 de CA: se muestra en la Figura 6.

2.3 Construcción de la Tabla EQM (Modelo de Calidad Extendido o Extended Quality Model); se adapta de las Tablas 10, 11 TraT de cada producto realizadas en la Fase 1.

La tabla EQM es otra forma de representar CA considerando propiedades no funcionales de calidad requeridas por cada componente funcional y posibles restricciones. Se deriva de las tablas TraT y puede ser utilizada para construir una representación ontológica de RA a efectos del proceso de derivación de productos en el ciclo de Ingeniería de la Aplicación [6] [30]; no será mostrada aquí para abreviar la presentación.

2.4 Actualización de CA con nuevos componentes.

Generalmente los nuevos componentes se refieren a propiedades no funcionales o requisitos de calidad que no se consideraron en la unión de lar representaciones gráficas de los productos; en este caso éstos son satisfechos globalmente por el uso de las plataformas, por lo tanto no se actualiza CA.

2.5 Actualización de las tablas IdAct e IdActGlob; en este caso tampoco hay actualización.

Salida para el sub-dominio TransaMóvil – Fase 2: $CA_{Transaμóvil:}$ Figura 6, EQMransaMóvil: tabla, IdActPWN/INT: Tabla 10, IdActGasMóvil :Tabla 11; IdActGlobTransaμóvil :Tabla 12.

C. Fase 3. Diseño del Dominio – sub-dominio TransaMovil - Figura 3

Entrada: $CA_{Transaμóvil}$: Figura 6, $EQM_{Transaμóvil}$: tabla, $IdAct_{Transaμóvil}$: Tabla 9;

Actividades:

3.1 Construcción de la RA TransaMóvil

3.3.1 Determinar puntos de variación, los cuales agrupan componentes que desempeñan tareas similares y podrán ser "instanciados" en RA para generar un producto concreto de la familia de productos del dominio.
En el caso de TransaMóvil, los puntos de variación, identificados en UML por el estereotipo *<<nombre del punto de variación>>*, son *<<a11.Portal>>, <<a12. Conect>>*, solo presente en PWN/INT y *<<b30. PagComb>>*, *<<b31. GesTrans>>*, *<<b32. Geoloc>>*, *<<b.33. RasPI>>* solo presentes en GasMóvil (ver Figura 7). Los demás componentes y sub-componentes de RA son comunes y constituyen el "core" o núcleo de componentes que siempre estarán presentes en cualquier producto de TransaMóvil, ver Figura 3. Debe recordarse que los puntos de variación [20] son componentes arquitecturales abstractas representadas como conjuntos de componentes variantes (no comunes), las cuales pueden ser seleccionadas cuando, al derivar un producto concreto, se instancia el punto de variación.
3.3.2. Construcción de la representación UML de la RA de TransaMóvil.

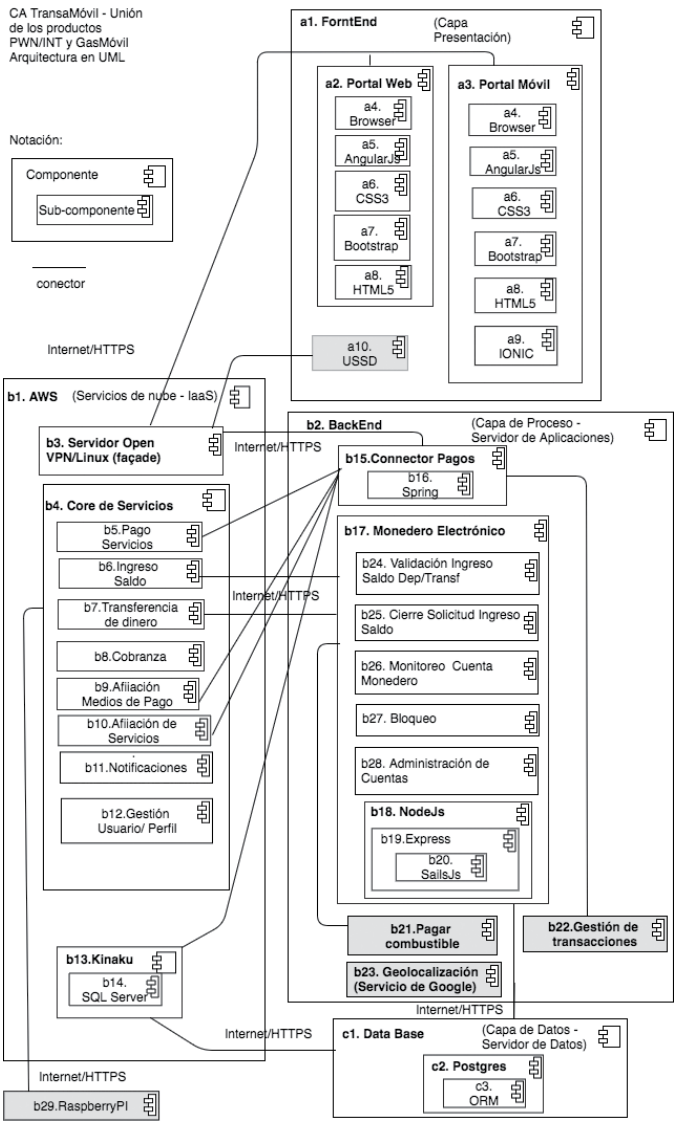

Fig. 6. Vista lógica UML de la arquitectura CA TransaMóvil: los componentes variantes se resaltan en gris. Fuente: autores

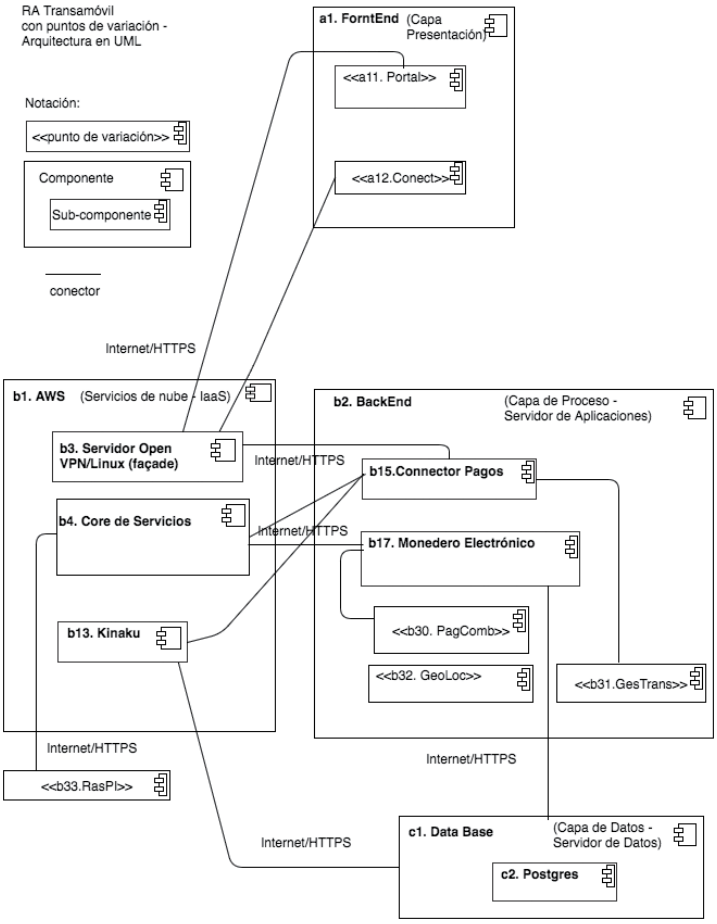

Fig. 7. Vista lógica UML de la arquitectura RA TransaMóvil. Fuente: autores

<<a11. Portal>> = {a2. Portal Web, a3. Portal Móvil}
<<a12. Conect>> = {a10. USSD}
<<b30. PagComb>> = {b21. Pagar Combustible}
<<b31. GesTrans>> = {b22. Gestión de transacciones}
<<b32. Geoloc>> = {b23. Geolocalización}
<<b.33. RasPI>> = {b29. RaspberryPI}

Se observa que para TransaMóvil solo se tienen dos variantes como alternativas de portales, sin embargo como la RA es una arquitectura genérica y evolutiva, si aparecen nuevas alternativas, dependiendo del avance tecnológico, éstas pueden ser agregadas sin problemas a los respectivos puntos de variación. Nótese que los productos existentes, como por ejemplo GasMóvil, pueden ser derivados de la RA

de TransaMóvil: *GasMóvil = {a1, a2, b1, b2, b3, b4, b15, b13, b17, b21, b22, b23, b29, c1, c2}*, los sub-componentes también están incluidas en esta configuración, integradas a su respectivo componente.

3.2 Elaboración del Documento de Recomendaciones Generales (DRG): análisis de los resultados obtenidos, limitaciones y recomendaciones para posibles mejoras.

3.2.1 Análisis de los resultados (Documento DRG para TransaMóvil):

Se ha aplicado IDCM (adaptado de y [9] [19]), el cual es un proceso sistemático y repetible en cuanto a su especificación, por lo tanto es también un activo o artefacto reutilizable por la empresa.

- Se han obtenido las arquitecturas en UML 2.0 (vista lógica de componentes y conectores [4]) de los productos desarrollados por TransaMóvil, mediante un proceso de reingeniería relativamente ágil *[9][19], estudiando la documentación existente (no el código), utilizando como instrumentos un quiz y tres reuniones con el equipo de trabajo.*

- Se han identificado los componentes similares (funcionalidades desarrolladas y plataformas, toolkits, protocolos etc.) utilizados, en cuanto a las tareas que realizan y sus conexiones (Tablas 6, 7, 8 CCT).- Se determinó la trazabilidad entre componentes funcionales y los requisitos de calidad por ellas exigidos, para satisfacer la calidad global de la RA, y garantizar así su carácter evolutivo, ver Tablas 10, 11 TraT y EQM (no mostrada en esta presentación).

*- Se observó que en el caso de la Computación Móvil y en particular de TransaM*óvil, las plataformas de desarrollo y los toolkits utilizados influyen en garantizar las propiedades de calidad que la aplicación desarrollada debe cumplir. Por lo tanto este resultado puede ser utilizado con fines publicitarios de los productos de la empresa, como una "certificación" de que los productos desarrollados son de "calidad" respecto al dominio considerado.

*- Se construyó la Tabla 9 IdAct de activos comunes y variables, que constituye el Repositorio de Activos de TransaM*óvil, objetivo principal del proyecto.

- Se construyó la RA TransaMóvil (ver Figura 9). Esta arquitectura es genérica, por lo tanto instanciable; a partir de ella se pueden generar variantes de productos o nuevos productos concretos eliminando componentes existentes, reutilizando o agregando nuevos componentes, tal como corresponde a una arquitectura que evoluciona con los cambios tecnológicos.

3.2.2 Limitaciones del proceso IDCM:

- La Fase 1. Alcance es la que resultó ser de mayor esfuerzo de trabajo, como se había previsto.

- Se observó que la documentación sobre la vista lógica de la arquitectura no existía, teniéndose solo diagramas de despliegue que únicamente muestran

la distribución física de las máquinas donde corre el sistema, pero no los componentes con sus conexiones, que son los que pueden reutilizarse como activos. Sin embargo, a partir de las arquitecturas especificadas en UML, se pueden realizar fácilmente los diagramas de despliegue asignando los componentes arquitectónicos a los respectivas servidores y máquinas, consiguiéndose así diagramas de despliegue más realistas.

- Se realizó un proceso de Ingeniería del Dominio completo. Faltaría realizar el proceso completo de Ingeniería de la Aplicación para mostrar en detalles como debe instanciarse la RA para la generación de nuevos productos, pero está fuera del alcance de este trabajo.

3.2.3 Recomendaciones

- RA y las Tablas 10, 11 IdAct y EQM deben ser actualizadas si hay cambios, para que este trabajo sea realmente de utilidad en la práctica usual de trabajo.

- La Tabla 12 IdActGlob representa el Repositorio de Activos, debería ser implementado como un verdadero repositorio con enlaces entre componentes arquitectónicos y módulos (código), fuera del alcance de este trabajo. Una representación ontológica de RA facilitaría el diseño del Repositorio de Activos [30].

- Faltaría especificar una 4ta. fase de IDCM para detallar el proceso de diseño de la RA del consorcio completo, en caso de tener varios sub-dominios.

- Sería recomendable como una buena práctica general, utilizar estándares tanto en la documentación como en los productos para el desarrollo, en todos los sub-dominios de la empresa, para facilitar la comunicación y la reutilización de material entre los equipos de trabajo.

Salida para el sub-dominio TransaMóvil – Fase 3: $CA_{TransaMóvil:}$ Figura 6, $EQM_{TransaMóvil}$, $IdAct_{PWN/INT}$: Tabla 10, IdActGasMóvil: Tabla 11, $IdActGlob_{TransaMóvil}$: Tabla 12, $RA_{TransaMóvil}$: Figura 7, $DRG_{TransaMóvil}$: texto. Los artefactos CA, EQM, IdAct, e IdActGlob son actualizados en la Fase 2, de ser el caso.

IV.2 Aplicación a los sub-dominios Sinapsis y Conectium

Se procede de la misma forma para los demás sub-dominios, aplicando el proceso IDCM completo obteniéndose una RA para cada sub-dominio, no se muestran aquí para abreviar la presentación. El proceso IDCM solo considera la construcción de las RA para cada sub-dominio y el repositorio global de activos IdActGlob (ver Tabla 12). Las tablas TraT para los sub-dominios (solo se mostraron las Tablas 10 y 11 para TransaMóvil), contienen la especificación de las propiedades de calidad requeridas y quienes las proporcionan. Fusionando las tres RA de los subdominios

considerado, obtenidas por IDCM y a partir de las tablas TraT, EQM e IdActGlob, se podría obtener la RA CONECTIUM para todo el consorcio; sin embargo este trabajo está fuera del alcance de la presente investigación, en vista de que la fusión de varios sub-dominios en una única RA es un proceso cuya especificación se encuentra aún en estudio y será objeto de trabajos futuros.

A continuación en la Tabla 12 se presenta uno de los resultados más relevantes del trabajo, el Repositorio de Activos completo IdActGlob del consorcio, con componentes comunes y variantes obtenido agregando las tablas $IdAcT_{TransaMóvil}$, IdAcTSinapsis y la IdAcTConectium .

Tabla 12. IdActGlob: Repositorio de Activos de CONECTIUM Limited C.A., Venezuela, para los sub-dominios TransaMóvil, Sinapsis y Conectium

Capa	Componentes comunes y variantes para el consorcio CONECTIUM	Comentarios
Capa Presentación (FrontEnd)	a1. FrontEnd	- FrontEnd para todos los productos
	a2. Portal Web	Portal; usado en los tres sub-dominios TransaMóvil, Sinapsis (producto WebApp) y Conectium
	- a4. Browser (común)	- Navegador, común
	- a5. AngularJs (TransaMóvil)	- Framework abierto de JavaScript – maneja MVC
	- a6. CSS3 (TransaMóvil)	- Estilo en cascada para doc. HTML
	- a7. Bootstrap (TransaMóvil)	- Framework abierto de desarrollo - maneja CSS y HTML
	- a8. HTML5 (TransaMóvil)	- lenguaje de marcas versión 5
	a3. Portal Móvil	Portal; usado en Transaóvil (productos PWN/INT y Gas Móvil para aplicaciones híbridas
	- a4. Browser (común)	- Navegador, comun
	- a5. AngularJs (TransaMóvil)	- Framework abierto de JavaScript – maneja MVC
	- a6. CSS3 (TransaMóvil)	- Estilo en cascada para doc. HTML
	- a7. Bootstrap (TransaMóvil)	- Framework abierto de desarrollo - maneja CSS y HTML - lenguaje de marcas versión 5 - para la portabilidad a diferentes SO

Continúa...

	- a8. HTML5 (TransaMóvil) - a9. IONIC (TransaMóvil) a10. USSD (TransaMóvil) a13. IU (Sinapsis) - a14. Java-Android (Sinapsis) - a15. Swift-iOS (Sinapsis) - a30. Portal AMC-Web (Conectium) - a4. Browser (común) - a34. JSP (Conectium) - a35. LDAP (Conectium) - a39. Login (Conectium) - a31. Portal Móvil-Web (Conectium) - a32. Portal AMC-MODELOS (Conectium) - a33. Portal ADMIN-CONTENIDOS (Conectium) - a36. SIMBROWSING (Conectium) - a37. Portal AMC-CONECTIUM (Conectium) - a38. Portal Móvil ALMANO-WAP (Conectium) - a42. Trivias SMS (Conectium) - Protocolos HTTP/HTTPS (comunes)	- conector; usado en producto PWN/INT - UI de escritorio de Sinapsis para ap. nativa NatApp; se baja de App Store (iOS) o Paly Store (Android); - portabilidad a Android, lenguaje Java (TransaMóvil) - portabilidad a iOS, lenguaje Swift (TransaMóvil) - Navegador - Lenguaje similar a PHP, pero usa el lenguaje Java; tecnología que ayuda a crear páginas web dinámicas basadas en HTML, XML; la usabilidad depende del diseño del portal realizado por Conectium - LDAP (Lightweight Directory Access Protocol) protocolo de acceso a directorios - componente de acceso a portales; no está presente en los portales tipo Móvil-Web - subcomponentes a4, a34, a35 - subcomponentes igual que a30 - subcomponentes igual que a30 - aplicación Java, utiliza MVC, para el FrontEnd; instalado en los Sym Cards de los teléfonos - subcomponentes igual que a30 - subcomponentes igual que a31 - proceso que se ejecuta constantemente para escuchar SMS, procesarlos y enviar respuesta - Protocolos de comunicación de red

Continúa...

Capa Proceso (Backend)	b1. AWS - servicios de nube - IaaS (común) b3. Servidor Open VPN/Linux (façade) (TransaMóvil) b4. Core de Servicios (TransaMóvil) - b5. Pago de Servicios - b6. Ingreso saldo - b7. Trasferencia de dinero - b8. Cobranza - b9. Afiliación Medios de Pagos - b10. Afiliación de Servicios - b11. Notificaciones - b12. Gestión Usuario/Perfil b13. Kinacu (TransaMóvil) - b14. SQL Server b15. Conector Pagos (TransaMóvil) - b16. Spring b17. Monedero Electrónico (TransaMóvil) - b21. Pagar Combustible - b22. Gestión de Transacciones - b23. Geolocalización - b24. Verif. Ingreso Saldo Dep/Transf. - b25. Cierre Solic. Ingreso Salldo - b26. Monitoreo Cuenta Monedero - b27. Bloqueo - b28. Administración de cuentas - b18. NodeJs - b19. Express - b20. SailsJs b34. Ruby-on-Rails (Sinapsis)	BackEnd para todos los productos - Façade para conectar capas de presentación (FrontEnd) y proceso (BackEnd) En la RA del consorcio CONNECTIUM solo se considerarán como variantes los componentes b3, b4, b13, b15, b17 con sus sub-componentes respectivos Servicios en b1. AWS utilizados por los productos PWN/INT y GasMóvil - Plataforma transaccional utilizada por los productos PWN/INT y GasMóvil - Framework abierto para el desarrollo de aplicaciones Web y gestión de transacciones para Java; similar a un EJB - funcionalidad de GasMóvil - funcionalidad de GasMóvil - funcionalidad de GasMóvil - Framework abierto para desarrollo bajo Java; contiene Express quien usa SailsJs; diseño robusto, maneja MVC - Plataforma de desarrollo utilizada por los productos WebApp y NatApp; es parte de b1. AWS; engloba todos los componentes y sub-componentes del BackEnd de Sinapsis

Continúa...

	- b35. Paciente (Sinapsis) - b36. Gestión de Paciente-Registro - b37. Control de Citas y Turno - b38. Manejo de HCE[17] (Sinapsis)	- Funcionalidades relativa a la gestión del paciente - Funcionalidades relativas a la atención al paciente - Funcionalidad relativa a la gestión de las HCE
	- b39. Practica Medica (Sinapsis) - b40. Doctores-Registro - b41. Consultas - b42. Medicamentos y Récipes	- Funcionalidades relativas a la práctica médica de atención al paciente presencial u “on-line”
	- b43. Informes Médicos - b44. Administración (Sinapsis) - b45. Facturación - b46. Laboratorios - b47. Aseguradoras	- Funcionalidades de tipo administrativo, incluye informes de resultados de laboratorios y relación con las aseguradoras
	b48. Seguridad (Sinapsis)	- Políticas de control de acceso a información medica, según tipo de personal
	b49. Info Lab (Sinapsis)	- Módulo externo para uso de laboratorios
	b60. Reportes (Conectium) - b74. Control de Acceso - b75. Cálculo	- Para proveedores y administrador
	b61. Gestión Usuarios (Conectium)	- maneja perfiles de usuarios; sub-componente b74
	b62. Ayuda (Conectium)	- helpdesk
	b63. Configuración de Billings (Conectium)	- facturación; sub-componente b74
	b64. Carga de Contenidos (Conectium)	- Conectium: carga de contenidos ya configurados; sub-componente b74
	b65. IMG (Conectium) - b66. cmtm-img-cliente	- maneja los servicios SMS con las operadoras de celulares
	b67. CPG (WS) (Conectium)	- Web Service; cobro a los usuarios suscritos de los servicios que se estén vendiendo en la operadora
	b68. Validación de suscripciones (WS) (Conectium)	- Web Service; valida suscripciones de usuarios corre cada hora del día para validar si existe configurado algún servicio de suscripción
	b70. Alertas SMS (Conectium) - b71. Scripts de Envío de Alertas	
	b72. Java (Conectium) b73. Groovy (Conectium) - Protocolos HTTP/HTTPS (comunes)	- plataforma del lenguaje Java para desarrollo - lenguaje similar a Java - protocolos de comunicación de red

Continúa...

[13] Historias Clínicas Electrónicas

Capa Datos (BackEnd)	- c1. SMBD (Común) - c2. Postgres (común) - c3. ORM o similares (variante) - c5. Hibernate (variante)	c1, c2 componentes comunes en la Capa de Datos - base de datos relacional - Mecanismo de conversión a objetos - Mecanismo de conversión a objetos

Respecto al documento DRG, presentado para TransaMóvil en la Sección IV.1, podemos agregar el hecho de que los desarrollos de la filial Conectium fueron los primeros realizados por el consorcio y no aplican las mismas estrategias de desarrollos de las otras filiales: en Conectium se utiliza solo programación Java y se implementan implícitamente en sus módulos muchos de los requisitos de calidad, que en las otras filiales son resueltos por mecanismos, plataformas o toolkits, de manera que no se pudo comprobar el cumplimiento de estas propiedades en las reuniones efectuadas. Por lo tanto en la RA de la filial Conectium se incluyeron nuevos componentes para satisfacer los requisitos de calidad que no se observaron explícitamente como resueltos, realizandose una actualización de CA. Como fruto de las discusiones en las reuniones, se detectó que realmente la implementación de estos requisitos deberían estar incluidos, pero para detectarlos habría que hacer una trabajo de reingeniería a nivel de código fuente, lo cual no estaba contemplado en el proyecto.

Conclusiones

En un contexto de ILPS se ha definido el proceso IDCM de Ingeniería del Dominio, inspirado en [9] para la estrategia ascendente de construcción de la arquitectura candidata y en [3] [19] para la fase de Alcance, para diseñar una RA en el dominio de la Computación Móvil. IDCM se ha aplicado paso a paso a un caso de estudio real realizado durante seis meses en un consorcio de desarrollo de software móvil, para tres sub-dominios específicos: transacciones financieras, sistemas de salud y contenidos de entretenimiento, para los cuales se han diseñado las respectivas RA. Nuestro proceso es sistemático, repetible y especificado semi-formalmente con fases, actividades y artefactos producidos. Se centra en una estrategia de reingeniería bottom-up extractiva, porque se estudian las arquitecturas de productos existentes en la empresa a partir de un cuestionario y entrevistas con los responsable del grupo de trabajo para identificar los principales componentes de los productos desarrollados. Las arquitecturas se representan en UML como una vista lógica de componentes y conectores, por grafos conexos no dirigidos y se realiza automáticamente una unión de grafos para determinar el núcleo de componentes comunes y variantes; se establece una correspondencia entre los componentes arquitectónicos, con una clara trazabilidad entre componentes que

resuelven funcionalidades y componentes que les proporcionan las propiedades de calidad requeridas, para garantizar así el carácter evolutivo exigido a una RA. Uno de los principales resultados, además de reconstruir las arquitecturas de los productos desarrollados, que no existían en la empresa sino a nivel de pocos diagramas de despliegue, fue de especificar el repositorio de activos reutilizables para cada sub-dominio y para el dominio general manejado por la empresa. Una observación importante en el desarrollo para la Computación Móvil, es que las propiedades de calidad, que deben ser satisfechas temprano para aumentar la flexibilidad a los cambios de la RA, son resueltas en cierta medida por el uso de toolkits, plataformas, frameworks y mecanismos, que garantizan la satisfacción automática de estas propiedades. Algunas limitaciones del proceso: la necesidad de tener herramientas computarizadas de soporte, ahora en vía de desarrollo, en vista de la complejidad del número de componentes que pueden aparecer en la práctica; para este estudio se manejaron alrededor de 70 componentes a un alto nivel de abstracción y no a nivel de código fuente. Una de las perspectivas interesantes para un futuro cercano, es seguir un enfoque ontológico para especificar las RA, en vista de que en ella se captura el conocimiento del dominio, para así validar su consistencia respecto a restricciones que pueden afectan sus componentes y realizar el proceso de derivación de productos en el ciclo IA [6] [30]. El enfoque ontológico permitirá también establecer un modelo conceptual para el diseño del repositorio de activos, que actualmente solo está especificado como una tabla de componentes; los activos para ser componentes realmente reutilizables deben estar diseñados con una interfaz que permita su interconexión y adaptar estos activos para la reutilización sería otra perspectiva interesante. En trabajos futuros habrá que considerar una cuarta fase en el proceso IDCM, para diseñar la RA que integre varios sub-dominios de un dominio dado. Pensamos que la aplicación paso a paso de IDCM a este caso de estudio real, extraido de las prácticas comunes de una industria de desarrollo de software móvil, será una guía útil para que el proceso pueda ser utilizado también en otros dominios o para desarrollar completamente una LPS en el dominio la Comutación Móvil.

Reconocimiento

Al consorcio CONECTIUM Limited C.A., Venezuela por habernos facilitado la realizaciónde este estudio y permitirnos divulgar sus principales resultados y a los líderes de los grupos de trabajo consultados Aquilino Pinto (TransaMóvil), Carlos Alonzo (Sinapsis) y Mary Gutiérrez (Conectium).

Referencias

Clements P. and Northrop L. (2001) SPL: practices and patterns, 3rd ed. Readings, MA, Addison Wesley.

Reinhartz-Berger I., A. Sturm, T. Clark, S. Cohen & J. Bettin, (Editors). (2013) Domain Engineering. Product Lines, Languages and Conceptual Models, Springer.

ISO/IEC NP 26550 (2013) Software and Systems Engineering – Reference Model for Software and Systems Product Lines. ISO/IEC JTC1/SC7 WG4.

M. Shaw M., Garlan D. (1996). Software Architecture. Perspectives of an emerging discipline, Prentice-Hall.

Object Management Group (OMG) (2005) Unified Modelling Language Superstructure, version 2.0 (formal/05-07-04), August.

Losavio F., Ordaz O., Jean S., (2016) Ontological approach to derive product configurations from a Software Product Line Reference Architecture, Revista Ciencia y Tecnología (CyT), UP, Argentina, Nº 16, pp. 91-127, ISSN 1850-0870, http://www.palermo.edu/ingenieria/pdf2016/CyT_16_07.pdf

Abrahamsson, P. (2005) Keynote: Mobile software development - the business opportunity of today. Int. Conf. on Software Development, pp. 20-23. Reykjavik.

Wasserman A. (2010) Software engineering issues for mobile application development, FoSER '10, FSE/SDP on Future of Software Engineering Research, pp. 397-400.

Losavio F., Ordaz O., Esteller V. (2015) Refactoring-Based Design of Reference Architecture, RACCIS (Revista Antioqueña de las Ciencias Computacionales y la Ingeniería del Software), Vol 5, No 1, pp. 32-48, enero-junio 2015, ISSN 2248-7441. URL, http://www.fundacioniai.org/raccis

Berard E. (1992) Essays in OO Software Engineering, Prentice Hall, N.Y.

World Wide Web Consortium (2010). Mobile Web Application Best Practices, W3C Working Draft, http://www.w3.org/TR/mwabp/

Agile Alliance. (2001) Agile Software Development Manifesto. Retrieved from Manifesto for Agile Software Development: http://agilemanifesto.org/

Boehm, B., Turner, R. (2003) Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley.

Spataru A., (2010) Agile Development Methods for Mobile App, MSc These, University of Edinburg.

Kaleel B. S., Harishankar S. (2013) Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its Challenges, Reyrson University, Canada, CSTR. Paper 4,

http://digitalcommons.ryerson.ca/compsci_techrpts/4

Salinesi C., Mazo R., Djebbi O., Dia D. (2011) Constraints: The core of product line engineering, Research Challenges in Information Science (RCIS), Fifth International Conference on, 19-21 May, pp 1-10.

Matinlassi M. (2004) Comparison of software product line architecture design Methods: COPA, FAST FORM, KobrA and QADA, ICSE'04.

Falvo Jr V.. et al. (2014) Towards the Establishment of a Software Product Line for Mobile Learning Applications, SEKE 2014.

Losavio, F., Ordaz O., Santos I. (2015) Proceso de análisis del dominio ágil de sistemas integrados de salud en un contexto venezolano, Revista Venezolana de Información, Tecnología y Conocimiento, ENL@CE, Vol. 12 (1)101-134 , Enero-Abril, ISSN: 1690-7515, http://www.produccioncientifica.luz.edu.ve/index.php/enlace/index

Pohl K., Böckle G., van der Linden F. (2005) SPL engineering - foundations, principles, and techniques. Springer IXXVI.

Bjørner D. (2006) Software Engineering 3: Domains, Requirements, and Software Design, Texts in Theoretical Computer Science, Springer-Verlag, Berlin Heidelberg.

Abrahamsson, P. (2007) Agile Software Development of Mobile Information Systems. In Advanced Information Systems (pp. 1-4). Berlin: Springer.

Schwaber, K. (2004) Agile Project Management with Scrum. Microsoft Press.

Mansanet I., Fons J., Torres I., Pelechano V. (2011) GeMMINi: Prototipo de IU sobre multiples dispositivos. Una estrategia basadas en LPS y MDD, FAZ.

Czarnecki K., Hwan C., Kim P.,Trygve K. (2006) Feature Models are Views on Ontologies, SPIC 2006.

Gruber T. (1993) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Technical Report KSL 93-04, KS Laboratory, Stanford University.

Chung L., Nixon B. and E. Yu. (1995) Using non-functional requirements to systematically select among alternatives in architectural Design,1st Inter. Workshop on Architectures for Software Systems, pp.31-42, Seattle, Washington.

ISO/IEC 25010 (2011) Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models, ISO/IEC JTC1/SC7/WG6.

Koziolek, H., Weiss, R., Doppelhamer, J. (2009) Evolving industrial software architectures into a software product line: A case study. Lecture Notes in Computer Science 5581, pp. 177-193.

Losavio F., Ordaz O. (2016) Reference Architecture Representation by an Ontology for Healthcare Information Systems Software Product Line, SCTC 2016 / ISBN: 978-980-12-8407-9, pp. 20-32, Universidad Central de Venezuela, Caracas, Venezuela – 9-11 de Mayo, http://www.sctc.org.ve/memorias/SCTC2016/SCTC2016-p020-032.pdf