UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
FACULTAD DE INGENIERÍA

CONSTRUCCIÓN DE UN MODELO DE TRAZABILIDAD QUE APOYE LA EVOLUCIÓN DE UNA LÍNEA DE PRODUCTOS DE SOFTWARE

POR
SEBASTIÁN ANDRÉS VALENZUELA BUSTOS

PROYECTO DE TÍTULO PRESENTADO A LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN, PARA OPTAR AL TÍTULO ACADÉMICO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: PEDRO ROSSEL CID.

COMISIÓN INFORMANTE: MARIELLA GUTIÉRREZ.

CONCEPCIÓN CHILE, ENERO 2018

AGRADECIMIENTOS

En primer lugar, quiero agradecer a Dios, por escuchar mis plegarias, cuando le pedía ayuda para que me acompañara y me diera fuerzas para seguir desarrollando este proyecto.

A mis padres y hermana por apoyarme, comprenderme y alentarme en los momentos de desmotivación personal.

A mi novia, Valesca, quien fue un apoyo fundamental en este proceso. Además, sin ella no hubiese terminado en el tiempo estipulado.

A mi profesor guía Pedro Rossel, que me soportó en su oficina varios días a la semana para contestar mis preguntas, preguntas que le repetí hasta terminar este proyecto, por la paciencia que tuvo conmigo y por compartir su conocimiento sin cobrarme ni una sola UF.

Tabla de contenido

Índice de figuras

Índice de tablas

Resumen del proyecto

Para lograr una evolución eficiente de una línea de productos de software, es imprescindible haber definido en las etapas del proceso de desarrollo del software la trazabilidad, que es la habilidad de proveer información de seguimiento de los requisitos, el diseño, implementación y pruebas de un sistema.

Cuando ocurren cambios en el software, la trazabilidad hace que sea relativamente más fácil evaluar el impacto que los cambios podrían tener en otras partes del proceso de desarrollo, garantizando que todos los requisitos sean diseñados y que todos los diseños se codifiquen y se prueben.

En este proyecto de título se propone un modelo de trazabilidad que apoya la evolución de una línea de productos de software (LPS), con el objetivo de disminuir el tiempo y el esfuerzo que se requiere para realizar alguna modificación dentro de los artefactos de trazabilidad que componen una LPS. Para lograr esto, se han establecido una serie de objetivos específicos, los cuales se resumen en comprender todos los aspectos relacionados a una LPS, tales como el proceso para llevar a cabo su desarrollo, su proceso de evolución, y la trazabilidad necesaria para llevar a cabo su evolución.

Para la validación del modelo de trazabilidad, se utilizó un caso de estudio desarrollado en el proyecto de título que lleva por nombre “Mantención de una familia de productos en el dominio de administración de emergencias”, desarrollado por Ana Montero Cáceres, alumna de la Facultad de Ingeniería de la Universidad Católica de la Santísima Concepción. Como resultado de la validación, se concluye que el modelo de trazabilidad sí apoya la evolución de una LPS, definiendo claramente los roles asociados a la evolución de los artefactos del modelo y definiendo la trazabilidad asociada a la evolución, según los escenarios de evolución de inserción, modificación y eliminación.

Summary

In order to achieve an efficient evolution of a software products line, it is essential to have traceability defined in the stages of the software development process, which is the ability to provide information about the requirements, design, implementation and testing of software.

When software changes occur, traceability makes it relatively easier to assess the impact changes might have on other parts of the development process, ensures that all requirements are designed and that all designs are coded and tested.

This title project proposes a traceability model that supports the evolution of a software products line (LPS), with the objective of reducing the time and effort required to make some modification within the traceability artifacts that form an LPS. To achieve this, a series of specific objectives have been established, which are summarized in the understanding of all aspects related to a LPS, such as the process to carry out its development, its evolution process, and the traceability necessary to carry out evolution.

For the validation of the traceability model, it was used a case study developed in the title project called "Maintenance of a family of products in the domain of emergency management", developed by Ana Montero Cáceres, student of the Faculty of Engineering de la Universidad Católica de la Santísima Concepción. As a result of the validation, it is concluded that the traceability model does support the evolution of a LPS, clearly defining the roles associated with the evolution of the artifacts of the model and defining the traceability associated with evolution, according to the evolution scenarios: insertion, modification and elimination.

1. Introducción

Una línea de productos de software (LPS) es un conjunto de sistemas de software que comparten un conjunto de características, las cuales satisfacen las necesidades específicas de un dominio o un segmento de mercado en particular. Estos sistemas de software se desarrollan a partir de un sistema común de activos bases reutilizables de una manera preestablecida [4].

Para el desarrollo de una LPS se producen dos fases de desarrollo: la Ingeniería del dominio, que se centra en generar activos reutilizables que incorporan las similitudes y variabilidades del dominio, y la Ingeniería de la aplicación, que utiliza los activos del dominio para crear productos de software específicos, con el objetivo de abordar las distintas necesidades de los clientes [1, 2].

Los principales beneficios del desarrollo de una LPS son: un menor tiempo y costo de desarrollo de los productos particulares, una disminución en el esfuerzo requerido para desarrollar un producto particular y un aumento en la calidad de los productos que la conforman [1].

Para lograr una presencia permanente en el mercado de sistemas de software, se requiere de una evolución constante en las características que conforman los productos de la LPS [2]. La evolución de una LPS se refiere a las modificaciones que se realizan en la LPS a lo largo del tiempo, con el objetivo de seguir satisfaciendo las necesidades de los clientes. La evolución de una LPS es un gran desafío, debido a la evolución independiente de los productos, los activos base y sus interacciones [3]. En palabras simples, si se desea realizar, por ejemplo, una modificación, eliminación o inserción de una nueva característica en el sistema, será necesario conocer las implicancias que estos cambios tienen en toda la LPS. Esto se puede llevar a cabo con la trazabilidad. La trazabilidad es la capacidad de interrelacionar los distintos artefactos que componen

una Línea de productos de software, manteniendo los enlaces de trazabilidad a lo largo del tiempo, y dicha información puede responder a preguntas sobre el producto y su proceso de desarrollo. Lo más importante, es que los enlaces de trazabilidad pueden usarse para analizar cómo los cambios aplicados a los artefactos existentes pueden afectar a otros artefactos en los mismos niveles de abstracción o diferentes.

1.1. Justificación del proyecto

En la actualidad existen empresas tales como Nokia, Boeing, Toshiba, Philips, entre otras, que se enfocan en el desarrollo de una serie de productos software, que comparten un conjunto de características comunes y que se diferencian en otras características variables, denominadas Líneas de Productos de Software (LPS) [4].

La constante aparición de nuevas necesidades, implica que aparezcan nuevos requerimientos del software, lo que conlleva una evolución de este tipo de productos. Ahora bien, si algún producto requiere de un cambio o evolución, es probable que los demás productos dentro de la LPS también requieran dichas actualizaciones debido a que comparten características con otros módulos de desarrollo.

Para lograr la evolución correcta de la LPS es imprescindible haber definido en las etapas de desarrollo la trazabilidad; sin embargo, definir la trazabilidad de forma manual para una LPS es complejo, costoso, propenso a errores y se requiere de mucho tiempo. Por estas razones sería útil contar con un modelo de trazabilidad que apoye a la evolución de una LPS, de manera que sea más fácil para las partes interesadas saber qué es lo que se debe observar al momento de realizar alguna modificación en una LPS [5], es decir, qué artefactos se ven involucrados con la modificación, quienes son los encargados de realizar dicha modificación, etc.

1.2. Delimitación del proyecto

El presente proyecto está orientado a la creación de un modelo de trazabilidad que apoye a la evolución de una línea de producto de software y no de un producto individual. No se construirá una herramienta, si no que se construirá un modelo de trazabilidad para líneas de producto de software. En cuanto a las etapas del desarrollo de la línea de producto de software que abarcará esta investigación se encuentran: el análisis, diseño, implementación y pruebas. El dominio de aplicación no está restringido.

Este proyecto de investigación tiene una duración de dos semestres, correspondientes a un año académico.

No necesariamente se partirá construyendo un modelo desde cero, sino que se podría tomar uno que ya cumple con ciertos requisitos y adaptarlo.

1.3. Objetivos del proyecto

1.3.1. Objetivo general

Construir un modelo de trazabilidad que apoye la evolución de la línea de productos de software.

1.3.2. Objetivos específicos

1. Entender el proceso productivo en una línea de producto de software.
2. Entender concepto de trazabilidad en línea de producto de software.
3. Determinar posibles eventos de evolución que pueden afectar la trazabilidad en una línea de productos de software.
4. Crear modelo de trazabilidad.
5. Validar modelo de trazabilidad.

1.4. Metodología

1.4.1. Etapa de aprendizaje de una Línea de producto de software

En esta etapa se estudiarán todos los conceptos relacionados con la producción de una línea de productos de software. Se deberá conocer qué son los activos base (*core assets*) y cómo éstos se construyen para dar inicio al desarrollo de una línea de productos de software. Por último, se deberá entender cómo se llevará a cabo la gestión del desarrollo de una LPS.

1.4.2. Etapa de aprendizaje de trazabilidad y evolución

Luego del estudio del desarrollo de una LPS, será necesario entender el concepto de trazabilidad en una línea de productos de software. Esto significa conocer el impacto que tendrán las modificaciones del software, ya sea durante o después del desarrollo del mismo, y cómo esas modificaciones se debiesen llevar a cabo. La trazabilidad es fundamental a la hora de evolucionar una línea de productos de software, debido al seguimiento de los requisitos en todo su proceso de desarrollo. Por último, se deberá entender cuáles serían los elementos y las relaciones a tener en cuenta al momento de hacer trazabilidad en una LPS.

1.4.3. Construcción y validación del modelo

Luego del proceso de aprendizaje de los términos relevantes a la investigación descritos anteriormente, se avanzará en la definición del modelo, es decir, definir el tipo de modelo que se construirá, pudiendo ser un modelado de procesos o mapa conceptual, por ejemplo. Posteriormente, se definirán los roles, artefactos a utilizar y los elementos de entrada y salida del mismo si es que los hubiese.

Finalmente, será necesario mostrar que lo que se construyó es al menos aplicable a un caso en particular. Para esto está contemplado utilizar un caso de estudio, es decir, existe una línea de productos de software realizada anteriormente que se utilizará para el proceso

de validación. La LPS a considerar se denomina *Línea de producto de software de apoyo a Emergencias para bomberos* [33].

2. Marco teórico

2.1. Línea de productos de Software

La definición de LPS comúnmente aceptada procede de [6], donde se definen las líneas de productos de software como *"un conjunto de sistemas software, que comparten un conjunto de características, las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base de una manera preestablecida".*

De esta definición se rescatan cinco conceptos fundamentales de las líneas de productos de software [4]*:*

- *"…un conjunto de sistemas software…":* El enfoque de las LPS no es el desarrollo de un producto singular, sino más bien de un conjunto de productos. Lo esencial es determinar cuál será el dominio de este conjunto.
- *"…que comparten un conjunto común de características*…": Las características son aquellos aspectos que poseen los productos que ayudan tanto a diferenciarlos entre sí, como a conocer las similitudes entre ellos. Por ejemplo, los teléfonos celulares Nokia, se fabricaron compartiendo ciertas características, es decir, con la reutilización de componentes y que a la vez se diferenciaban en otras, a modo de satisfacer a cada segmento de clientes en particular.
- *"…satisfacen las necesidades específicas de un dominio o segmento particular de mercado…":* Las LPS se desarrollan pensando en un segmento de mercado concreto. Como se explicó anteriormente, los productos intentan satisfacer necesidades específicas de un segmento de mercado. De la correcta determinación de este segmento de mercado dependerá el éxito que obtendrá la LPS.
- *"…se desarrollan a partir de un sistema común de activos base…":* Las LPS se desarrollan a partir de una serie de elementos reutilizables. Estos activos base podrían ser una serie de elementos, tales como requisitos, planificaciones, modelo de características, arquitecturas, componentes, código fuente, entre otros. Todos

estos elementos serán la base sobre la que se construirá cada producto. El desafío no es solo determinar lo común, sino que también determinar lo que va a variar dentro de la serie de productos de la LPS.

- *"…de una manera preestablecida…":* La manera en que se construirán los productos está perfectamente establecida con anterioridad, a través de un plan de producción.

Estos cinco conceptos ayudan a entender los que es el desarrollo de las líneas de productos de software.

El objetivo principal de una línea de productos de software es desarrollar una familia de productos software similares a partir de un conjunto de activos base compartidos [7]. A diferencia de la ingeniería de productos singulares, LPS muestra una alta complejidad debido a la variabilidad e interdependencia entre los distintos productos que la componen.

Las LPS cumplen con el anhelo recurrente que posee la ingeniería de software, que es la reutilización. Ahora bien, se debe tener en cuenta que desarrollar y a la vez gestionar estos artefactos reusables conlleva un costo inicial mayor, debido a la Ingeniería del dominio que debe efectuarse para lograr el desarrollo de éstos.

El proceso de desarrollo de una LPS consiste en la fase de Ingeniería del dominio y la fase de Ingeniería de la aplicación.

La ingeniería del dominio aglutina todo el desarrollo de los elementos comunes de la LPS. El equipo de desarrollo tendrá como objetivo desarrollar los elementos del dominio, estudiar el dominio, definir el alcance dentro del mercado objetivo de la LPS, definir las características, implementar los elementos comunes reusables, su mecanismo de variabilidad, y por último establecer cómo será el plan de producción [8]. "Básicamente se centra en el desarrollo de los elementos reutilizables que formarán la familia de productos, identificando las partes comunes y variables de la familia" [4, 6].

La ingeniería de la aplicación toma en consideración todo el desarrollo del producto en particular. Sus objetivos incluyen desarrollar los productos para clientes concretos, a partir de los requisitos del cliente. “Se centra en el desarrollo de productos individuales pertenecientes a la familia de productos y que satisfacen un conjunto de requisitos y restricciones expresados por un usuario específico, reutilizando, integrando y adaptando los elementos reutilizables existentes producidos en la Ingeniería de Dominio [9].”

La figura 1 representa un modelo general de las fases de Ingeniería del dominio e Ingeniería de la aplicación, con las etapas correspondientes a cada una de las fases. Cada una de las etapas de la Ingeniería del dominio genera artefactos reutilizables, denominados artefactos del dominio. Algunos de estos artefactos reutilizables serán instanciados hacia las etapas de la Ingeniería de la aplicación y otros serán desarrollados o modificados para satisfacer alguna aplicación en particular. Cabe destacar que existirá una retroalimentación para cada una de las etapas de desarrollo de la LPS con el objetivo de entregar el mejor producto posible.

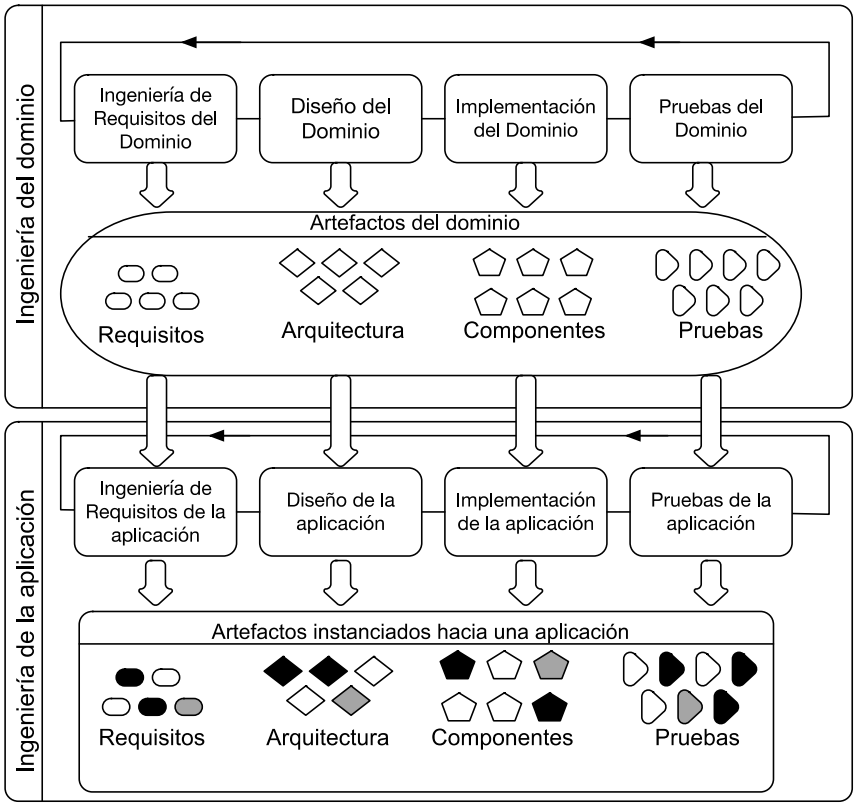

Figura 1: Etapas y artefactos relacionados a la Ingeniería del dominio e Ingeniería de la aplicación [24]

2.1.1. Beneficios relativos a la productividad y al costo

Las LPS pueden aumentar considerablemente la productividad de los Ingenieros de software. Esto se entiende como una reducción en el esfuerzo y el costo necesario para desarrollar cierta cantidad de productos software similares y a su vez darles mantenimiento [4].

En los casos de estudio se han observado mejoras en la productividad que duplican o triplican los enfoques tradicionales de desarrollo de software.

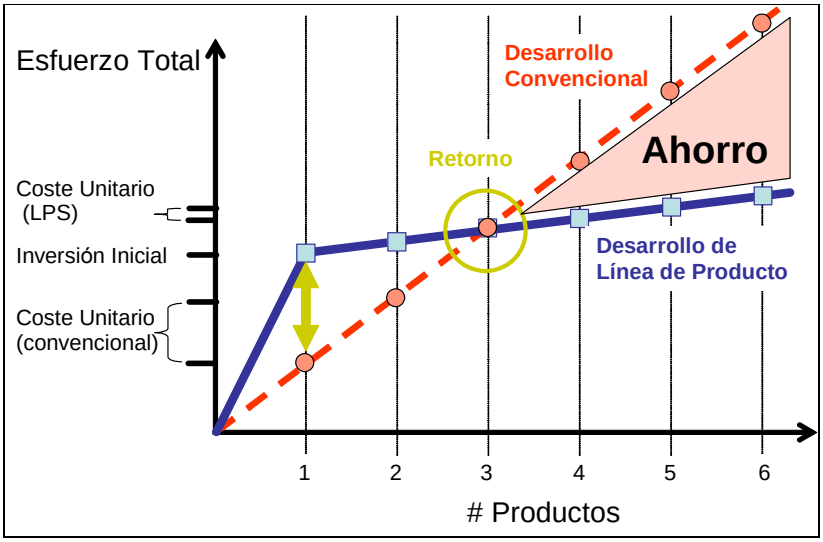

Figura 2: Desarrollo convencional vs. línea de productos [4]

La Figura 2 ilustra el esfuerzo y el costo necesario para desarrollar, poner en marcha y mantener un conjunto de productos de software similares. Para el enfoque tradicional no se obtienen ahorros importantes por el hecho de ser productos semejantes. El esfuerzo asociado al producto 6 es similar al esfuerzo que conlleva el producto 4. Cada producto evoluciona de manera independiente, y es posiblemente gestionado por equipos independientes.

Por el contrario, el enfoque de LPS solo al inicio se necesita de un mayor esfuerzo, pues se realiza un estudio inicial para analizar el dominio, determinar las variaciones a soportar, y desarrolla la plataforma de LPS. La intersección entre las dos líneas de la figura 2 muestra el punto a partir del cual se empieza a rentabilizar esta inversión inicial. A partir de aquí la generación de productos es menos costosa con la LPS que con el enfoque inicial. Sin embargo, si la familia de productos es poco numerosa (menor a 3), probablemente no se compensen los gastos iniciales de desarrollo de la plataforma.

En el desarrollo convencional, se crea un producto y luego se utiliza un enfoque muy común denominado "*copy&own*". Se realiza una copia de una aplicación previa que se parece a la actual, y a partir de aquí la aplicación evoluciona de manera totalmente independiente, es decir, con este enfoque utilizado el parecido de las aplicaciones sirve solamente para agilizar el desarrollo, pero en ningún caso toma en cuenta el mantenimiento. Si se desea realizar mantenimiento, tendrá que hacerse por separado, aunque estas aplicaciones sean muy similares.

En conclusión, el desarrollo convencional solo se centra en el producto, por lo que cada producto tiene su propio mantenimiento, por ende, no se aprovecha el beneficio potencial que se podría obtener derivado de la semejanza entre productos.

Por el contrario, el desarrollo basado en LPS consiste en gestionar tanto lo común de los productos, como lo variable. La reutilización es planificada y la incorporación de nuevas funcionalidades se hace de manera sistemática y controlada. Todos estos factores agilizan tanto el desarrollo, como el mantenimiento general de los productos y su puesta en el mercado. Además, un error encontrado en un producto, debiese ser fácilmente detectable en el resto de los productos que componen la LPS, gracias al enfoque de desarrollo basado en componentes reutilizables.

Esta reducción en el costo de mantener un producto, permite abarcar un mayor número de productos o de variantes dentro de la LPS. Esto a su vez trae ciertas ventajas estratégicas de mercado, por ejemplo: cubrir un mayor número de mercados con productos adaptados; crear un mayor número de mercado enfocado a ciertos segmentos específicos de mercado; incrementar la agilidad para expandirse a nuevos mercados y reducir el riesgo asociado a la puesta en marcha de los productos.

2.1.2. Beneficios relativos a la calidad

La calidad en una LPS se puede medir de dos formas [4].

- Primera, con el grado de precisión con que cada producto se ajusta a las necesidades del cliente. Esta medida depende del grado de variabilidad de la LPS. A mayor variabilidad, mayor es la probabilidad de adaptar el producto a los requerimientos del cliente. Como obviamente, el generar variabilidad tiene un costo de producción, el reto está en encontrar el equilibrio entre la variabilidad y ese costo.
- El segundo aspecto de calidad es la tasa de defectos de los productos dentro de la LPS. Aquí los beneficios se derivan de la reutilización de los elementos comunes (*core assets*). A mayor cantidad de productos, quiere decir que mejor será la calidad, ya que se asume que los productos están muy probados/depurados. Además, los beneficios de encontrar y eliminar un defecto en un *core assets*, no se limita solamente al producto en que se encontró el error, sino que a todos los productos de la LPS [4].

En la figura 3, se muestra la tendencia descendente de la cantidad de defectos a medida que aumenta la cantidad de productos de la LPS. Esto se debe a que los productos anteriores sirven como aprendizaje de errores para los productos posteriores en la LPS. Los errores son corregidos y retroalimentados para que no ocurran de nuevo.

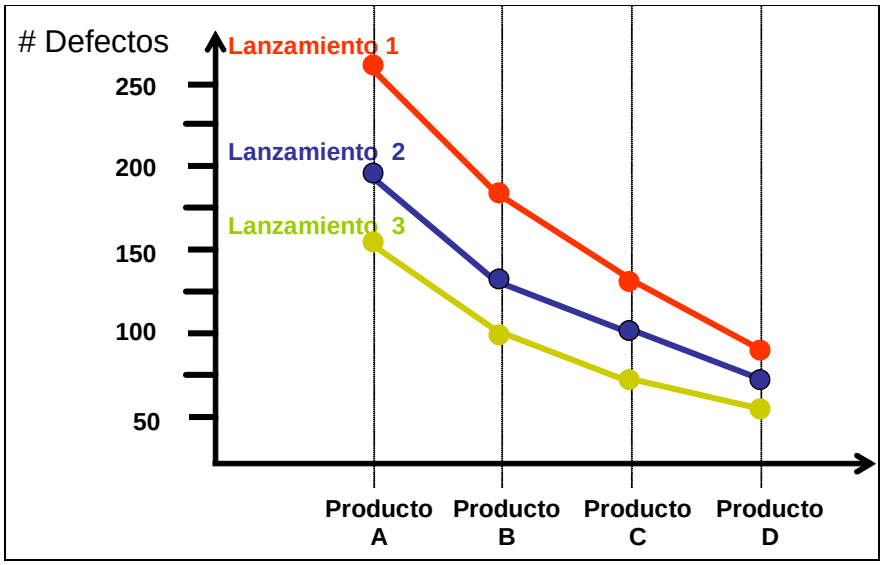

Figura 3: Defectos en LPS

2.2. Trazabilidad

2.2.1. Requisitos de trazabilidad

Conceptualmente cuando se habla de trazabilidad, se refiere al seguimiento de las relaciones entre los requerimientos que existen dentro del desarrollo del software, el cual es esencial, debido a la gran cantidad de información que es usada, producida y que debería estar relacionada entre sí.

Cuando se desarrollan grandes sistemas, es muy complejo recordar todos los enlaces que se crearon para relacionar la información. Incluso resulta imposible cuando se cuenta con un equipo multidisciplinario que se distribuye las tareas dentro del proyecto de desarrollo. Por lo general, se busca hacer un seguimiento a una cadena de enlaces, lo que va más allá del punto de partida del rastreo [10].

La trazabilidad de los requisitos se refiere a la capacidad de describir y hacer seguimiento a la vida de un requisito tanto en dirección hacia delante, como hacia atrás [11].

Un requisito es registrado en un documento porque se supone que debe ser trazado. Puede ser que el requisito se extraiga directamente del documento o puede ser que el documento contenga declaraciones de apoyo al requisito. En cualquier caso, la existencia de alguna de estas influencias de una sobre otra es necesaria si han de ser vistos como objetos relacionados [11] . Un requisito se traza hacia adelante a un componente de diseño para llevar el rastro del requisito. Nuevamente, este componente puede ser diseñado tanto para encontrar el requisito, como para realizar un procedimiento de testeo del mismo.

La siguiente definición reconoce las ocurrencias naturales de las trazas y el hecho de que hay que identificar cuáles trazas debiesen ser controladas, la forma en que deben ser registradas y el seguimiento posterior que se les debe realizar [12]:

“ Trazabilidad de requisitos se refiere a la habilidad para definir, capturar y seguir el rastro dejado por los requisitos en otros elementos relacionados al entorno de desarrollo de software y los rastros dejados por aquellos elementos sobre los requisitos” [10].

Está claro que el entorno de desarrollo de software implica no solo lo técnico, sino también los aspectos sociales del desarrollo del software. Sus elementos técnicos no son solo la especificación de requisitos, los diagramas, el código, sino también las personas, las políticas, las decisiones, incluso algunas cosas menos tangibles como los objetivos y las ideas.

2.2.2. Modos de Trazabilidad

Hay varias formas en las que se puede realizar el rastreo de los requisitos. En cuanto a la dirección del rastreo, un requisito puede ser rastreado hacia adelante o hacia atrás. En cuanto a la evolución de los requisitos, éstos pueden ser rastreados para aspectos ocurridos antes o después de su inclusión en la especificación de requisitos, y dependiendo del tipo de objetos involucrados, podrían verse varios artefactos involucrados en la trazabilidad.

2.2.2.1 Trazabilidad hacia adelante y hacia atrás

Estos dos conceptos serán ilustrados en la figura 4, la cual contiene definiciones estándares en la literatura [13]:

Trazabilidad hacia adelante, es la habilidad para rastrear un requisito para componentes del diseño o implementación.

Trazabilidad hacia atrás, es la habilidad para rastrear un requisito hasta su origen, es decir, hasta la persona, institución, la ley, argumentos, etc.

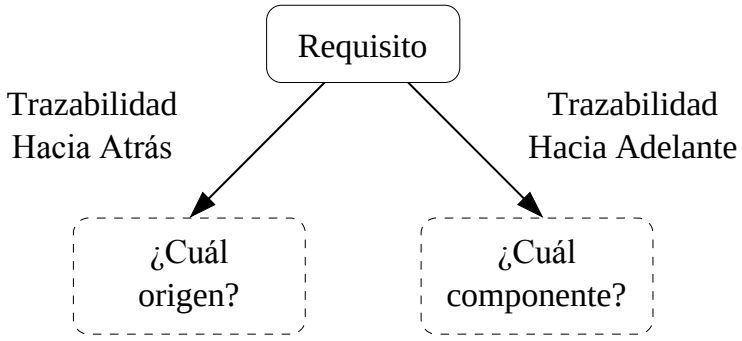

Figura 4: Trazabilidad hacia adelante y hacia atrás

Un requisito es trazado hacia adelante, por ejemplo, cuando ha sido modificado y se quiere conocer el impacto que ha producido el cambio. Se quiere obtener todo lo relacionado con los procedimientos de pruebas, para decidir cuáles de estas pruebas deberían ser añadidas, modificadas o eliminadas. Todo lo anterior sirve para asegurar que los cambios que se han realizado cumplan con los requisitos esperados. También es posible que se desee obtener los componentes construidos para cumplir con el requisito y analizar los cambios que se deben hacer en aquellos componentes.

Un requisito es trazado hacia atrás, por ejemplo, cuando ha ocurrido un cambio y se quiere saber de dónde proviene, investigando la información utilizada para obtener el requisito modificado. Se puede querer saber cuál es la persona interesada en el requisito, saber de qué documento específicamente se extrae el requisito, o saber a qué departamento de una organización está asociado [10].

2.2.3. Trazabilidad Pre y Post Especificación de Requisitos

La especificación de requisitos es el resultado de la información obtenida a través de entrevistas al cliente y la posterior deducción de los requisitos. El trazado de un requisito puede ser realizado por dos motivos: para obtener información acerca del proceso de obtención de la información, antes de su inclusión en la especificación de requisitos, o para

obtener información relacionada con su uso, después de que el requisito haya sido deducido e incluido en la especificación de requisitos. La figura 5 ilustra el concepto de trazabilidad previa a la especificación de requisitos y trazabilidad después de la especificación de requisitos.

2.2.3.1. Trazabilidad Pre-especificación de requisitos

Se refiere a aquellos aspectos en la vida de un requisito previo a su inclusión en la especificación de requisitos [10].

2.2.3.2. Trazabilidad post-especificación de requisitos

Se refiere a aquellos aspectos en la vida de un requisito que resultan de la inclusión en la especificación de requisitos [10].

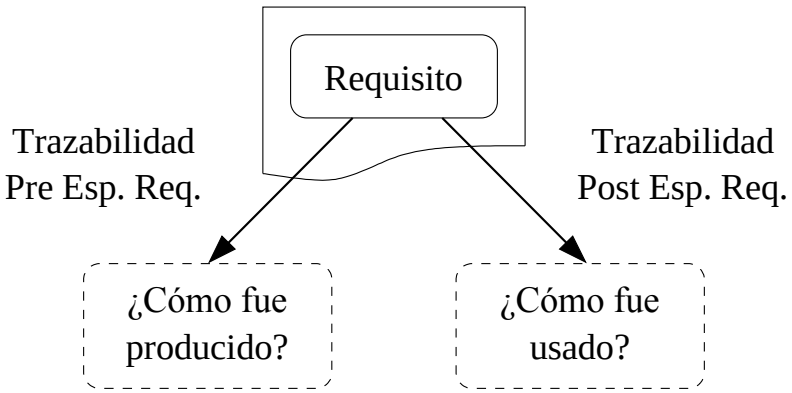

Figura 5: Trazabilidad pre y post especificación de requisitos

Trazabilidad pre-especificación de requisitos es útil, por ejemplo, cuando ha ocurrido un cambio en un requisito y se quiere obtener el origen del requisito o saber cuáles son las personas involucradas en la validación del cambio.

Trazabilidad post-Especificación de requisitos es útil, por ejemplo, para obtener el módulo de diseño en el cual fue alojado o los procedimientos de pruebas creado para verificar los requisitos.

2.2.4. Trazabilidad inter-requisitos y extra-requisitos

Hacer una especificación de requisitos conlleva una gran cantidad de trabajo, debido a la gran cantidad de detalles que ésta debe poseer, para ser entendida por personas que no son precisamente del área de trabajo. Esto conlleva a reformular y redefinir varias veces los requisitos hasta ser completamente entendidos. En efecto, requisitos nuevos aparecen derivados de los antiguos y también otros son desechados. La habilidad para rastrear las relaciones entre requisitos se denomina trazabilidad inter-requisitos. Las relaciones entre requisitos y otros elementos son capturados por la trazabilidad extra-requisitos.

La trazabilidad inter-requisitos es sobre todo importante a la hora de analizar y hacer frente a los requisitos que han sido evolucionados o modificados. Se hace trazabilidad inter-requisitos, por ejemplo, cuando se quieren conocer todos los requisitos que son derivados de otro requisito o su proceso de refinamiento.

2.3. Trazabilidad en Líneas de Productos de Software

Cabe recordar que el desarrollo de una línea de producto de software consta de dos fases: 1) Ingeniería del dominio y 2) Ingeniería de la aplicación, y a su vez cada fase posee cuatro etapas de desarrollo. Las cuatro etapas de la fase de la Ingeniería del dominio son:

Ingeniería de requisitos del dominio, Diseño del dominio, Implementación del dominio y Pruebas del dominio. Las cuatro etapas de la fase de la Ingeniería de la aplicación son: Ingeniería de requisitos de la aplicación, Diseño de la aplicación, Implementación de la aplicación y Pruebas de la aplicación. Cada etapa de desarrollo mencionada anteriormente entrega como resultado los artefactos de la trazabilidad, estos pueden ser: documentos, características, requisitos, código fuente, arquitectura de referencia, informe de pruebas, entre otros. Estos artefactos pueden ser modificados, eliminados o creados conforme aumentan las necesidades de las partes interesadas en la LPS. Además, cada uno de los artefactos puede estar relacionado con otros artefactos, ya sean de una misma etapa, de diferentes etapas, o de diferentes fases. Con lo mencionado anteriormente se definen tres enfoques de trazabilidad en una LPS, éstas son:

- Trazabilidad interna: se refiere a la relación que existe entre los artefactos de una misma etapa de desarrollo, por ejemplo, algunos de los artefactos de la Ingeniería de requisitos del dominio se relacionan con otros artefactos de la misma etapa (ver figura 6).
- Trazabilidad horizontal: se refiere a la relación que existe entre los artefactos de distintas etapas, pero de una misma fase, por ejemplo, algunos de los artefactos de la etapa de Ingeniería del dominio se relacionan con los artefactos de la etapa de Diseño del dominio (ver figura 6).
- Trazabilidad vertical: se refiere a la relación que existe entre los artefactos de distintas fases, por ejemplo, los artefactos de la etapa de Ingeniería de requisitos del dominio se relacionan con los artefactos de la etapa de Ingeniería de requisitos de la aplicación (ver figura 6).

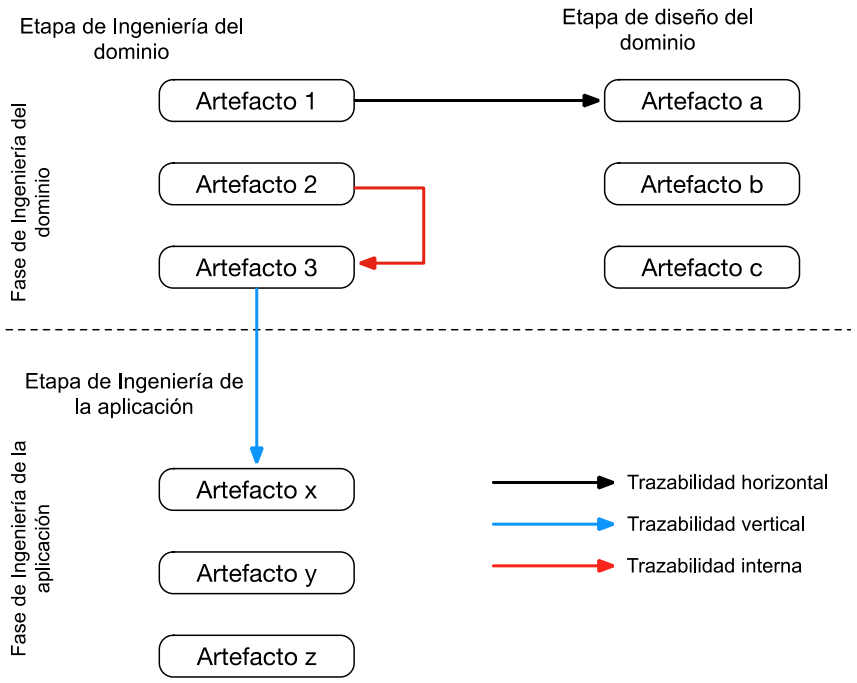

Figura 6: Ejemplo de trazabilidad vertical, horizontal e interna

2.4. Evolución en línea de producto de software

Las LPS son una entidad de larga vida útil y deben evolucionar para satisfacer los nuevos requisitos por muchos años. Sin embargo, es imposible prever todos los cambios y, en consecuencia, el alcance de la LPS desde el principio. Los cambios pueden ser introducidos, por ejemplo, por las nuevas tecnologías, preferencias del consumidor o los competidores.

Existe un consenso general de que, en el contexto de líneas de productos de software, la evolución es un aspecto importante. Sin embargo, la gestión estratégica de la evolución de una línea de productos no ha sido ampliamente cubierta por la literatura. En cambio, la mayoría de los investigadores informan sobre los enfoques para apoyar cambios en la implementación, mientras que la evolución es llevada sobre la marcha, es decir, la línea de

productos se extiende caso a caso para satisfacer los requisitos de los nuevos productos [14].

La evolución de las LPS tiene severos desafíos causados por las características que poseen las LPS [7]:

- Larga vida útil: por un lado, LPS es una inversión a largo plazo que vale la pena en cuanto a más productos se deriven de la LPS. Por otra parte, una LPS debe evolucionar para reflejar los nuevos requisitos modificados para sus productos. Por lo tanto, una línea de productos de software evoluciona por más tiempo y en mayor medida que los productos de software individuales.
- Gran tamaño y complejidad: LPS representa toda la familia de productos, esto conlleva una mayor complejidad que la de un producto individual. Usualmente múltiples equipos de desarrollo se unen para la creación y mantención de ésta.
- Mayor interdependencia: debido a la reutilización sistemática en una LPS, existe mayor interdependencia entre los activos del software. Por ejemplo, los cambios realizados a nivel de LPS, pueden afectar a muchos productos individuales que se encuentran en la LPS, y nuevos requisitos a nivel de producto individual, pueden provocar cambios en toda la LPS.

Para analizar el impacto de los cambios a raíz de la evolución, los activos de una LPS se pueden clasificar en tres categorías [7]:

- *Activos comunes:* son definidos en la LPS como parte de todos los productos, por tanto, quedan directamente derivados de la LPS. Como ejemplo, en la figura 6 estos activos corresponden a MP3, Llamadas y SMS.

- *Activos variables:* son definidos en la LPS como parte de algunos productos, dependiendo de la configuración de cada producto. Por lo tanto, a nivel de cada producto, tiene que haber una decisión acerca de la variabilidad del activo variable,

la cual es impulsada por los requisitos para el producto en particular. Como ejemplo, en la figura 7 estos activos corresponden a 3G, Video, Bluetooth y Juegos.

- *Activos específicos del producto:* son usados para añadir funcionalidad a cada producto individual, por lo que algunas funciones añadidas específicas de cada cliente no son compatibles con los activos reutilizables de la LPS. Por lo tanto, los activos específicos del producto y sus requisitos correspondientes residen solo a nivel de producto y no hay ninguna configuración de variabilidad para ellos. El uso de los activos específicos del producto idealmente debe reducirse al mínimo dentro de un enfoque de LPS, debido a que disminuye la reutilización y aumenta el esfuerzo para el mantenimiento. Sin embargo, dependiendo del mercado y del modelo de negocios, no siempre es posible rechazar los requisitos específicos del producto. Como ejemplo, en la figura 7 este activo corresponde a Java.

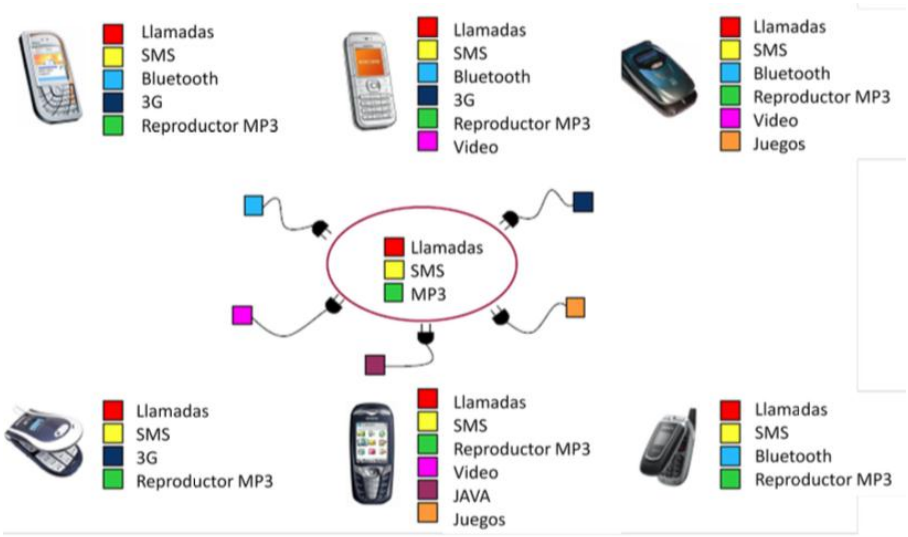

Figura 7: Ejemplo de activos comunes, variables y específicos del producto

En cuanto al conjunto de productos, nuevos productos pueden ser añadidos, así como también otros en desuso se pueden desechar.

En cuanto a los activos específicos del producto, éstos pueden ser añadidos, eliminados o modificados. Los cambios deben ser propagados hacia niveles más bajos de abstracción (ejemplo, desde requerimientos hacia implementación). Los cambios en los activos comunes se realizan a nivel de LPS y afectan a todos los productos que la componen. Los cambios en los activos variables que se realizan a nivel de LPS, afectan a todos los productos en los que se seleccionan las variantes respectivas.

2.4.1. Análisis del impacto del cambio

Cuando se decide optar por algún cambio, se tiene que predecir el esfuerzo requerido y peligros potenciales para su realización. Esto es soportado por el enfoque del análisis de los cambios.

Un aspecto importante a la hora de analizar el impacto es la trazabilidad, ya que es importante conocer los enlaces entre todos los activos lógicos relacionados en el proceso de desarrollo de software, esto se hace para entender qué otros activos pudiesen ser afectados si un activo cambia. Un ejemplo son las trazas entre los requisitos y sus activos de implementación. En el contexto de LPS, el mapeo entre las características y los activos de implementación puede ser considerado como un tipo de enlace de traza. Ver figura 8.

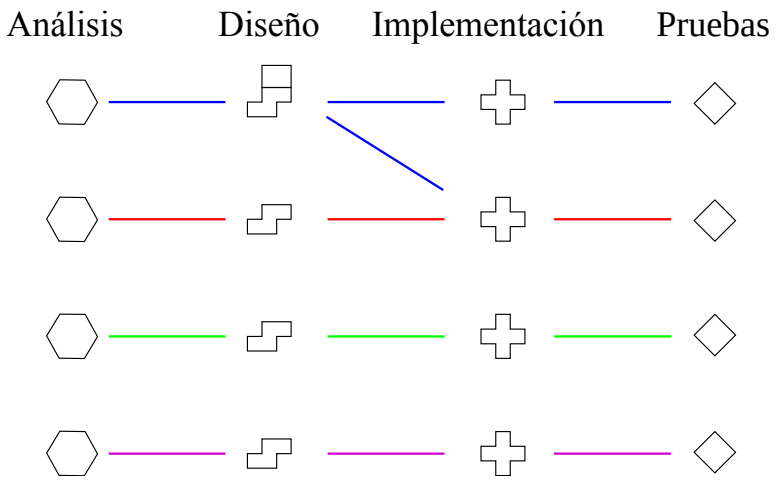

Figura 8: Trazabilidad de requisitos

En la figura 8 se observa que, al agregar un componente de diseño, se debe buscar otro componente dentro del producto que satisfaga dicha necesidad. Por lo tanto, se genera una nueva traza entre un componente del diseño y un componente de implementación. Este procedimiento se debe realizar cada vez que se desee implementar, eliminar o modificar algún componente, ya sea de cualquier etapa de desarrollo de la LPS.

2.5. Modelo

Un modelo es una representación simplificada de la realidad [15]. No precisamente contiene todos los elementos de la realidad, sino que aquellos que se consideran más importantes para lograr una representación visual de algún tema que se quiera explicar.

Un modelo debe tener un propósito bien definido, que a su vez debe resolver un problema en particular. Contar con un propósito bien definido es el éxito para lograr un exitoso modelo de estudio [15].

Cada modelo es una vista auto contenida del sistema [16], es decir, el usuario de un modelo no necesita más información de otro modelo para interpretarlo. Por consiguiente, debiese haber solo una interpretación de lo que ocurrirá en el sistema cuando ocurra un evento descrito previamente en el modelo. Por otra parte, un modelo debe describir las interacciones entre el sistema y lo que le rodea.

La utilidad de un modelo puede tener los siguientes matices [17]:

- Ayuda para aclarar el pensamiento acerca de un área de interés.
- Sirve como una ilustración del concepto.
- Ofrece una contribución para definir estructura y lógica.
- Se constituye en un prerrequisito al diseño.
- Refleja los aspectos esenciales del objeto o fenómeno de forma simplificada.

- Optimiza la actividad práctica mediante la transformación de la realidad.

Existen dos tipos de modelos; modelos cualitativos y modelos cuantitativos.

2.5.1. Modelos cualitativos

Los modelos cualitativos determinan de manera general, las relaciones que existen entre diferentes factores o elementos del sistema. El objetivo principal de este modelo es facilitar el entendimiento de cómo funciona el proceso especifico de interés [18]. Al construir modelos gráficos, se aconseja comenzar de manera sencilla para luego ir ampliando el modelo hasta lograr incluir todos los elementos esenciales del modelo.

2.5.2. Modelos cuantitativos

Una vez que se desarrolla el modelo cualitativo que represente la realidad, se puede proceder a incluir números y expresiones matemáticas para así convertirlo en un modelo cuantitativo. Principalmente, esto se realiza para refinar aún más el modelo conceptual, ya que se intenta incluir valores numéricos a todos los elementos incluidos en el modelo [18].

2.6. Modelos de trazabilidad

"Los métodos de desarrollo de software son variados y tienen características propias que los hacen aptos y específicos para las necesidades de los desarrolladores. Sin embargo, independientemente de cual se utilice y los productos de trabajo o artefactos que de él se deriven, los elementos que apoyan el proceso de desarrollo son susceptibles a ser trazados". Ahora bien, el grado de trazabilidad depende de ciertos factores tales como la cantidad y la

calidad de información que proporcionan los elementos del modelo y las necesidades de los participantes del proyecto en la gestión que se deriva de la traza [19].

Los elementos de trazabilidad reconocen tres elementos básicos [19]: las partes interesadas (*stakeholders*), las fuentes (documentos y modelos) y los objetos o artefactos para ser trazados. Estos elementos y su evolución se deben identificar explícitamente en cada flujo de trabajo para así controlar y soportar el trazado en las fases del proceso. Los modelos de trazabilidad se deben generar por iteración para que los grupos de trabajo tomen decisiones acerca del alcance del desarrollo y del impacto del cambio.

3. Estado del Arte

3.1. Subprocesos y artefactos relacionados para la fase de Ingeniería del domino y la fase de Ingeniería de la aplicación

Los autores (Van der Linden, Böckle y Klaus, 2005) desarrollaron un marco de referencia para la Ingeniería de línea de productos de software, el cual consiste en la incorporación de los conceptos fundamentales de la Ingeniería de línea de productos de software, el uso de plataformas y la capacidad de proporcionar una personalización masiva.

Una plataforma es, en el contexto del software, un conjunto de artefactos reutilizables, los que deben ser utilizados de manera consistente y sistemática para construir aplicaciones, es decir, que se puedan incorporar con un orden establecido siguiendo un método en particular. Los artefactos reutilizables abarcan todo tipo de artefactos de desarrollo del software, tales como modelos de requisitos, modelos arquitectónicos, componentes del software, planes de prueba, diseños de pruebas, entre otros.

Para facilitar la personalización masiva, la plataforma debe proporcionar los medios para satisfacer los diferentes requisitos de las partes interesadas. Para este propósito, se introduce el concepto de variabilidad en la plataforma, es decir, que los artefactos pueden variar de acuerdo a las aplicaciones de la línea de productos, dependiendo del tipo de clientes.

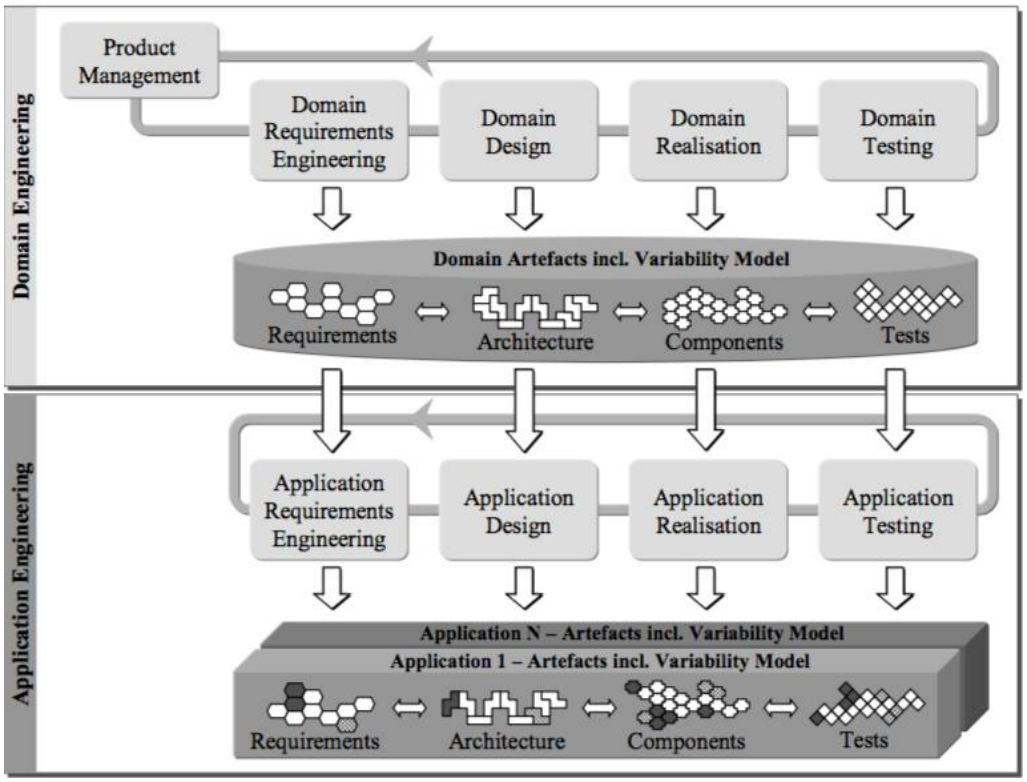

Figura 9: Marco de referencia para la Ingeniería de línea de productos de software

La figura 9 muestra el marco de referencia para la Ingeniería de línea de productos de software, en la cual se observan las dos fases de desarrollo de una LPS, es decir, la fase de Ingeniería del dominio y la fase de Ingeniería de la aplicación. Además, cada fase se divide en cuatro etapas de desarrollo. La fase de Ingeniería del dominio se divide en: Ingeniería de requisitos del dominio, Diseño del dominio, Aplicación del dominio y Pruebas del dominio, y la fase de Ingeniería de la aplicación se divide en: Ingeniería de requisitos de la aplicación, Diseño de la aplicación, Implementación de la aplicación y Pruebas de la aplicación. Cada etapa de la Ingeniería del dominio genera como resultado los artefactos reutilizables del dominio, y cada etapa de la Ingeniería de la aplicación utiliza los artefactos de la Ingeniería del dominio para construir aplicaciones particulares. Cabe destacar que en la fase de Ingeniería de la aplicación se pueden modificar o incluir artefactos que no existían en la Ingeniería del dominio (tal como se muestra en la parte inferior de la figura 9,

en donde se aprecia que no todos los artefactos son de color blanco, debido a las modificaciones que se realizaron a los artefactos).

Por otra parte, se define que cada etapa de desarrollo tiene una relación directa con las etapas que se encuentran antes, después y debajo de la misma, por ejemplo, la etapa de Diseño del dominio se relaciona con las etapas de Ingeniería de requisitos del dominio (al lado izquierdo), Implementación del dominio (al lado derecho) y Diseño de aplicación (debajo de Diseño del dominio), tal como se muestra en la figura 10.

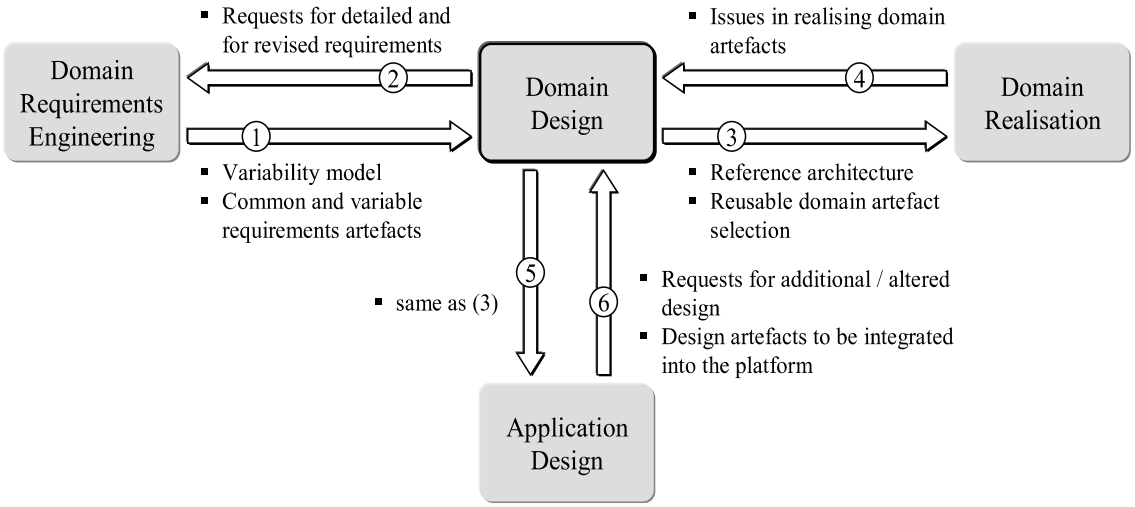

Figura 10: Flujos de información entre la etapa de Diseño del dominio y otras etapas de desarrollo

Como se aprecia en la figura 10, cada etapa envía información a la siguiente y recibe a su vez una retroalimentación para mejorar aquellas falencias que se encontraron. La información enviada de una etapa a otra consiste en los artefactos reutilizables que fueron desarrollados por cada etapa. Por ejemplo, la etapa de Ingeniería de requisitos del dominio entrega a la etapa de Diseño del dominio dos artefactos, que son: el modelo de variabilidad y los artefactos de requisitos comunes y variables.

3.1.1. Conclusiones sobre el marco de referencia propuesto

El marco de referencia propuesto por los autores Van der Linden, Böckle y Klaus en [20], se utilizará de base para la construcción del Modelo de trazabilidad que se desarrollará en este proyecto, ya que posee las cuatro etapas de desarrollo de una LPS, tanto para la fase de Ingeniería del dominio, como para la fase de Ingeniería de la aplicación. Además, considera la retroalimentación entre las etapas de desarrollo, con el objetivo de construir productos de software de calidad.

Por otra parte, este framework incluye los artefactos reutilizables como resultado de cada etapa, considerando que cada artefacto de salida será el artefacto de entrada para la etapa siguiente.

La principal diferencia con el modelo que se desarrollará en este proyecto, es que este framework no es un modelo de trazabilidad, por lo tanto, no incluye la trazabilidad directa entre los distintos artefactos reutilizables. Sin embargo, entrega un aporte importante en cuanto a las definiciones de los artefactos reutilizables, y de los roles asociados a cada artefacto.

4. Construcción del modelo de trazabilidad

El desarrollo del modelo de trazabilidad propuesto en este proyecto se basa en el modelo de la figura 11, obtenida de [20]. Esta figura representa el modelo general del desarrollo de una línea de productos de software, en la cual se destacan los principales artefactos reutilizables del dominio construidos para cada etapa de la Ingeniería del dominio. Posteriormente, se eligen algunos de los artefactos reutilizables construidos en las etapas de la Ingeniería del dominio para instanciarlos en las etapas de la Ingeniería de la aplicación (artefactos destacados en color negro). Pueden existir artefactos en las etapas de la Ingeniería de la aplicación que deban ser modificados con el objetivo de cumplir alguna necesidad específica de un producto (artefactos destacados en color plomo). Finalmente, para lograr un resultado óptimo en el desarrollo general de una línea de productos de software, existirá retroalimentación entre las distintas etapas, tanto en la Ingeniería del dominio como en la Ingeniería de la aplicación.

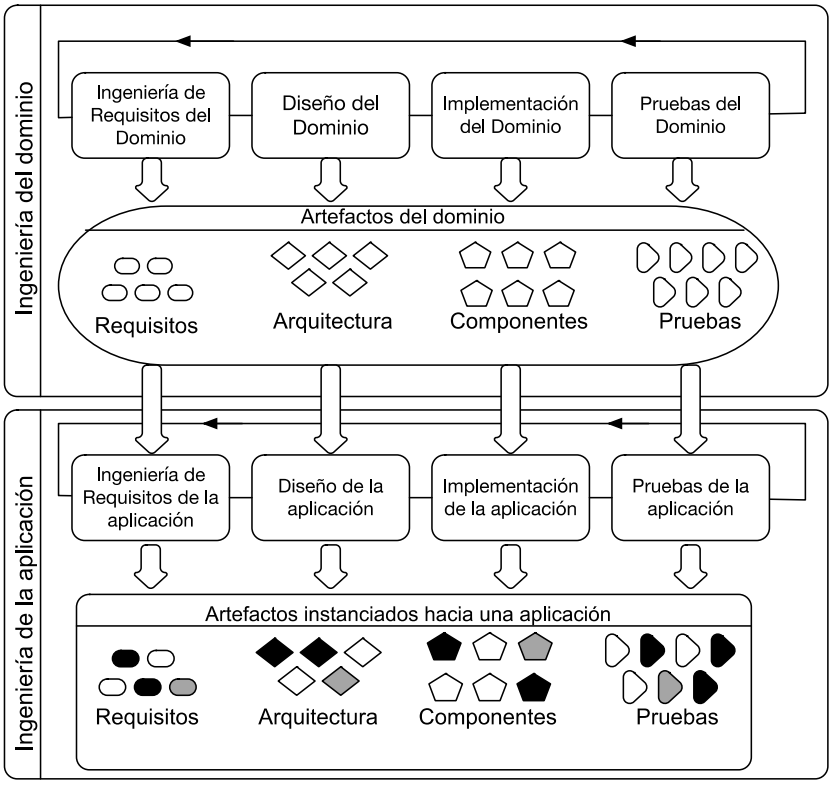

Figura 11: Modelo general del desarrollo de una línea de productos de software

Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

4.1. Artefactos para la fase de Ingeniería del dominio

4.1.1. Etapa de Ingeniería de requisitos del dominio

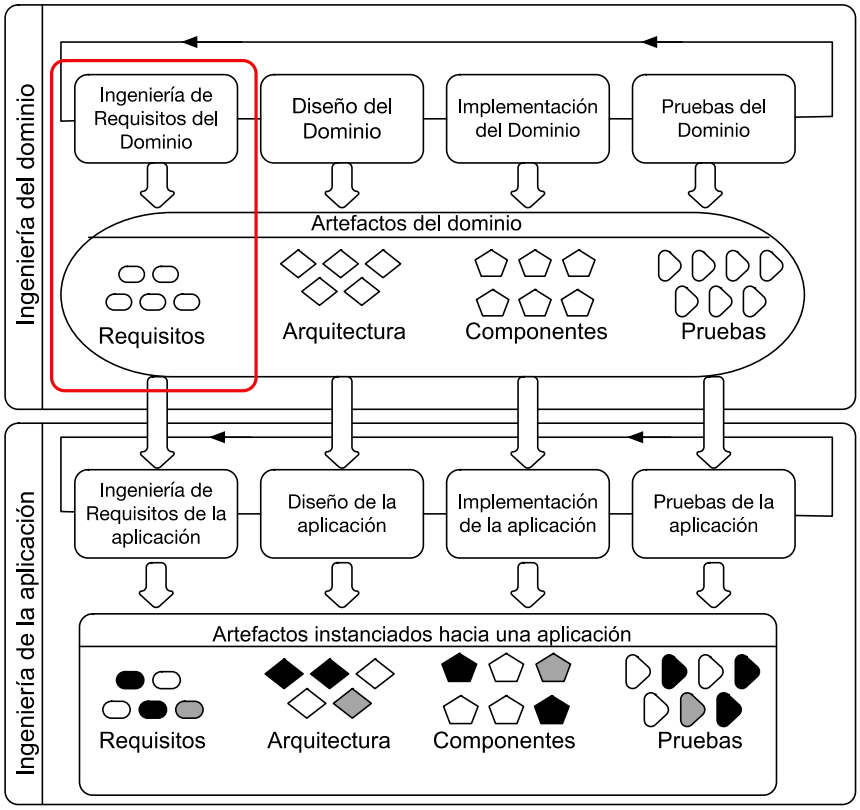

Figura 12: Etapa de Ingeniería de requisitos del dominio y sus artefactos

Se definen a continuación los artefactos del dominio correspondientes a la etapa resaltada en la figura 12, es decir, la etapa de Ingeniería de requisitos del dominio.

- Alcance: El Alcance (*scope*) define todos los posibles productos que abarcará la LPS. Dentro del Alcance se encuentran: el modelo de características, los objetivos y el mapa de productos [21]. Se considera como artefacto debido a que consolida el conocimiento sobre el dominio que se necesita para iniciar otras actividades en la

LPS [21] y además es considerado como un activo base importante dentro de la LPS [21].

- Modelo de Características: Es el artefacto comúnmente usado para modelar el dominio [21] y expresa las posibles combinaciones de características que pueden tener los productos dentro de la LPS [22]. Este debe identificar los puntos en común y variables dentro de la LPS [21].
  - Característica (*feature*): es la característica visible para el usuario final [22], por ende se transforma en uno de los artefactos más influyentes dentro de esta etapa.
    - Características comunes: Son aquellas características que se encuentran en todos los productos de la LPS [21].
    - Características variables: Son aquellas características que se encuentran solo en algunos de los productos de la LPS [21].
  - Restricciones (*constraints*): Este artefacto representa las características que posee un producto, ya sea de forma obligatoria (que debe poseer dicha característica), opcional (pudiese poseer que dicha característica) o excluyente (al poseer una característica, no podrá poseer la otra) [21].
  - Relaciones: Este artefacto es el que une a un producto con las restricciones en cuanto a las características que tenga dicho producto.
- Objetivos: Este artefacto define los objetivos que el sistema considerado debe alcanzar de acuerdo a las personas interesadas de la LPS [23].
- Mapa de productos: Este artefacto describe un conjunto de productos finales que pertenecen a la LPS y las características que éstos poseen, ya sean características comunes o características variables [20].

- Caso de Uso: Este artefacto representa una interacción de uno o más actores (usuario y/o sistema) con el sistema considerado. Además, provee una descripción del comportamiento del sistema en términos de escenarios que ilustran diferentes maneras de fallar o tener éxito en alcanzar uno o más objetivos [22].

- Requisitos: Este artefacto es considerado como el requerimiento del cliente o alguna otra parte interesada sobre el sistema [1]. Los requisitos más relevantes pasan a ser características del sistema, por lo que pasarán a formar parte del modelo de características [20].
    - Requisitos funcionales: Es la funcionalidad visible del sistema y describe cualquier actividad que este deba realizar, aunque en algunos casos los requisitos funcionales también pueden declarar lo que el sistema no debe hacer. Dichas funcionalidades pueden ser comunes para todos los productos de la LPS o pueden ser variables, es decir, que solo las utilicen algunos productos de la LPS [24].
    - Requisitos no funcionales: Se basa en las restricciones de los servicios o funciones ofrecidas por el sistema, es decir, son las propiedades emergentes de un producto de software, por el cual su calidad será juzgada por alguna parte interesada. También son conocidos como atributos de calidad, tales como los de rendimiento, seguridad, modificabilidad, confiabilidad, usabilidad, entre otros. Son considerados como artefactos ya que tienen una influencia significativa en la arquitectura de software de un sistema [21].

4.1.2. Diseño del Dominio

El objetivo principal de la etapa de Diseño del dominio es producir la arquitectura de referencia, definiendo la estructura principal del software y la textura [20].

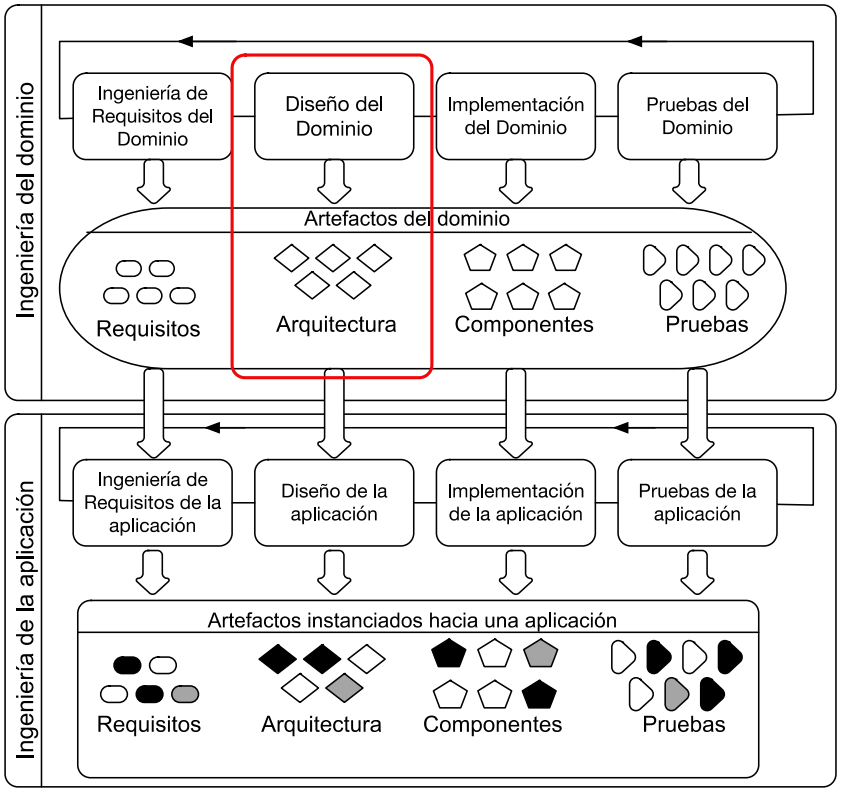

Figura 13: Etapa de Diseño del dominio y sus artefactos

Se definen a continuación los artefactos del dominio correspondientes a la etapa resaltada en la figura 13, es decir, la etapa de Diseño del dominio.

Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

- Requisitos Arquitectónicos: Como en la realidad los requisitos no están listos para ser diseñados a tiempo, los arquitectos toman los requisitos en evolución e identifican los más relevantes, los cuales conformarán la arquitectura de referencia [24].
- Arquitectura de Referencia: Es considerada como un artefacto ya que es la arquitectura central que captura el diseño de alto nivel para las aplicaciones de la línea de productos de software [20].

    - Estructura: Es la descomposición de un sistema en sus principales componentes y sus relaciones, y determina qué partes de construyen por separado [20].
    - Modelo de Comportamiento: Establece el comportamiento del sistema en base a los requisitos diseñados [24].
    - Textura: Este artefacto constituye una recopilación de las reglas comunes de desarrollo para la realización del sistema [25]; estas reglas pueden expresarse como convenciones de codificación, patrones de diseño y estilos arquitectónicos [24].
- Decisiones de Diseño: Son todas las decisiones sobre variabilidad en el diseño de los componentes de la estructura, las cuales deben ser comunicadas y documentadas para uso futuro [20].

4.1.3. Implementación del Dominio

El objetivo principal de la implementación de dominio es proporcionar el diseño detallado e implementación de activos de software reutilizables, basados en la arquitectura de referencia [20].

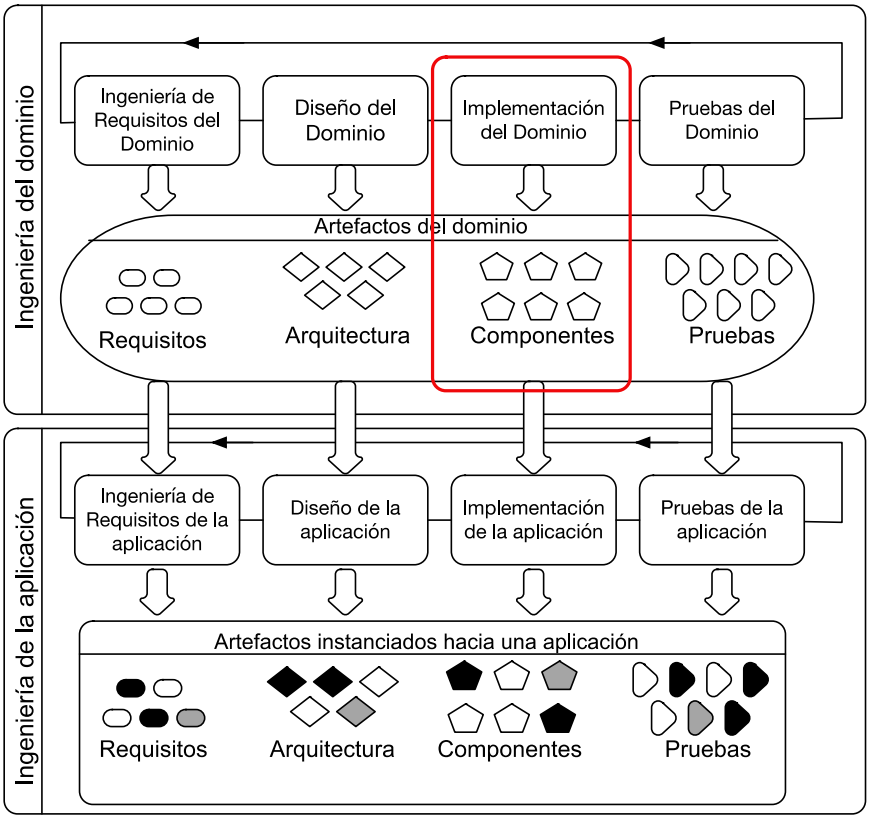

Figura 14: Etapa de Implementación del dominio y sus artefactos

Se definen a continuación los artefactos del dominio correspondientes a la etapa resaltada en la figura 14, es decir, la etapa de Implementación del dominio.

Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

- Componente Reutilizable: Este artefacto se construye a partir de la arquitectura de referencia y son las piezas fundamentales con las cuales se construyen las aplicaciones de la LPS [20]. Los Componentes Reutilizables están compuestos por los módulos de código fuente y bibliotecas.

- Interfaces: Son las partes externamente visibles de los componentes y son los medios para conectar los componentes. Los componentes proporcionan funcionalidad a otros componentes a través de una interfaz proporcionada [20].
- Generadores de Código: Este artefacto es el que permite enlazar los módulos de Código Fuente. Los Generadores de Código son construidos para la futura integración en productos específicos [21].
- Lenguaje de Dominio Específico: Este artefacto es el lenguaje de programación usualmente limitado a un lenguaje de dominio en particular [26], es decir, un lenguaje adaptado a un dominio de la aplicación específica. Por ejemplo, cobol para computación de negocios, fortran para computación científica, LaTeX para tipografía, entre otros.
- Documentación de Implementación: Este artefacto contiene la documentación asociada a todos los componentes reutilizables construidos en la etapa de Implementación de Dominio, el lenguaje de dominio específico que se debe usar y en el control de versiones en los activos reutilizables.

Los Componentes Reutilizables y los Generadores de Código permiten generar aplicaciones de forma automática o semiautomática, dependiendo de cuántos desarrollos personalizados sean necesarios [21].

4.1.4. Pruebas del Dominio

En esta etapa se validan y verifican los componentes reutilizables en función de sus especificaciones, es decir, requisitos, arquitectura y artefactos de implementación. Además, se desarrollan artefactos de prueba reutilizables para reducir el esfuerzo en las pruebas de aplicación [20].

Estos artefactos de prueba reutilizables son productos del proceso de testeo que contiene los planes, las especificaciones y los resultados de las pruebas [27].

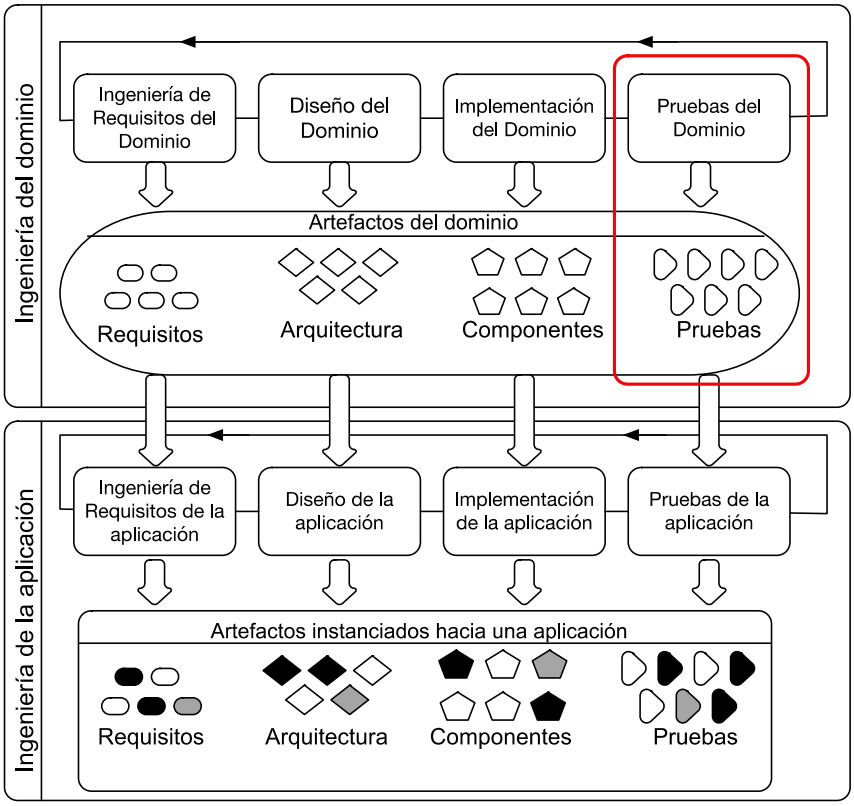

Figura 15: Etapa de Pruebas del dominio y sus artefactos

Se definen a continuación los artefactos del dominio correspondientes a la etapa resaltada en la figura 15, es decir, la etapa de Pruebas del dominio.

Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

- Plan de pruebas de dominio: Este artefacto define la estrategia de prueba para las pruebas de dominio, los artefactos de prueba a crear y los casos de prueba que se deben ejecutar. También define el programa y la asignación de recursos para las actividades de prueba de dominio [20].
- Resultados del Testeo: Este artefacto comprende los resultados de las pruebas realizadas en las pruebas de dominio y en las pruebas realizadas en los artefactos reutilizables del dominio [20].

- Casos de prueba de dominio: Este artefacto define las condiciones, los datos de entrada, y los datos de salida esperados para la prueba. Cada caso de prueba define un objetivo de prueba e incluye uno o varios escenarios de casos de prueba [27].
    - Escenarios de Caso de Prueba de Dominio: Este artefacto describe una secuencia específica de acciones a ejecutar. La ejecución de estos escenarios resulta en el logro de la meta de la prueba [27].

Tanto los casos de prueba de dominio y como los escenarios de caso de prueba proporcionan instrucciones detalladas para el Ingeniero de pruebas que realiza el test, y por lo tanto hacen las pruebas rastreables y repetibles.

4.2. Artefactos para la fase de la Ingeniería de la aplicación

4.2.1. Etapa de Ingeniería de requisitos de la aplicación

Es la selección de las características deseadas para un producto en particular [21]. Además, comprende tanto los requisitos reutilizados, como los requisitos específicos de la aplicación [20]. Pueden existir ciertas diferencias entre los requisitos de aplicación definidos por las partes interesadas y los requisitos del dominio definidos en la etapa de ingeniería del dominio, es decir, que los requisitos del dominio no satisfagan a todos los requisitos de la aplicación. En tales casos, los requisitos del dominio podrían ser adaptados para satisfacer algún requisito en particular. Los artefactos de la etapa de Ingeniería de requisitos de la aplicación comprenden todos los artefactos de desarrollo de una aplicación específica, incluyendo la propia aplicación configurada y probada [20]

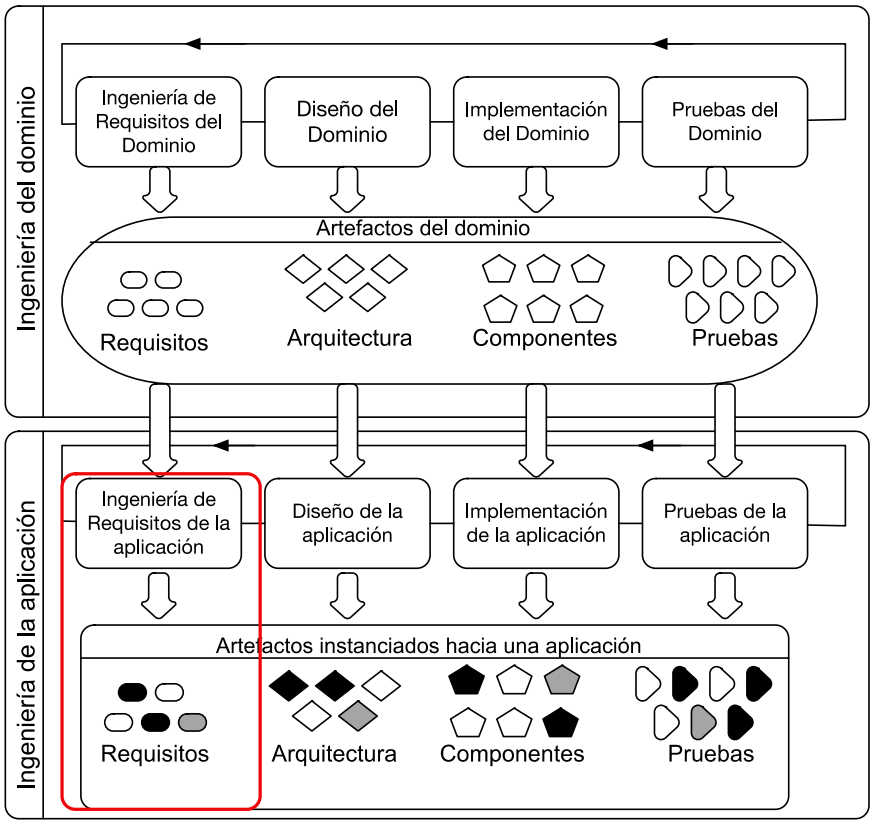

Figura 16: Etapa de la Ingeniería de requisitos de la aplicación y sus artefactos

Se definen a continuación los artefactos de la aplicación correspondientes a la etapa resaltada en la figura 16, es decir, la etapa de Ingeniería de requisitos de la aplicación.

- Requisitos particulares: Este artefacto comprende alguno de los requisitos desarrollados por la Ingeniería de requisitos del dominio, los cuales son elegidos para el desarrollo de un producto en particular [20].
    - Requisitos particulares funcionales: Este artefacto ya definido en el Análisis del dominio, comprende los requisitos funcionales instanciados a los productos particulares, es decir, se seleccionan los requisitos funcionales que cada producto comprenderá para ser construido.
    - Requisitos particulares no funcionales: Este artefacto ya definido en el análisis del dominio, comprende los requisitos no funcionales instanciados a

productos particulares, es decir, se seleccionan los requisitos no funcionales que cada producto comprenderá para ser construido.

- Requisitos específicos: Este artefacto comprende los requisitos recientemente desarrollados en la etapa de Ingeniería de requisitos de la aplicación o requisitos desarrollados en la etapa de Ingeniería de requisitos del dominio, pero que han sido adaptados para satisfacer un requerimiento en particular [20].
    - Requisitos específicos Funcionales: Dependiendo de los requisitos funcionales, el arquitecto selecciona las variantes del producto a entregar y luego construye el sistema ejecutable basado en esa selección [28].
    - Requisitos específicos No Funcionales: Los requisitos funcionales pueden ir acompañados de los no funcionales, los cuales darán a conocer el comportamiento que tendrán los requisitos dentro del sistema, además de ciertos atributos de calidad deseables [28].

- Configuración del modelo de características: Este artefacto contempla los elementos del Modelo de características que están presentes en un producto particular [21].
- Documentación de Requisitos de la aplicación: Este artefacto comprende la documentación de todos los artefactos de la etapa de Ingeniería de requisitos de la aplicación, incluyendo tanto los requisitos presentes en la etapa de Ingeniería de requisitos del dominio (requisitos particulares), como también los requisitos que se crearon en la etapa de Ingeniería de requisitos de la aplicación y que fueron aceptados por los partes interesadas (stakeholders) para ser incluidos en la LPS [21].

4.2.2. Etapa de Diseño de la aplicación

Define la arquitectura del producto único para el producto en particular [21]. Las entradas para esta etapa son la arquitectura de la línea de productos, la cual es proporcionada por la etapa de Diseño del dominio y la especificación de requisitos del dominio, la cual es proporcionada por la etapa de requisitos de la aplicación. La salida de esta etapa es la Arquitectura de la aplicación para el producto en particular [21].

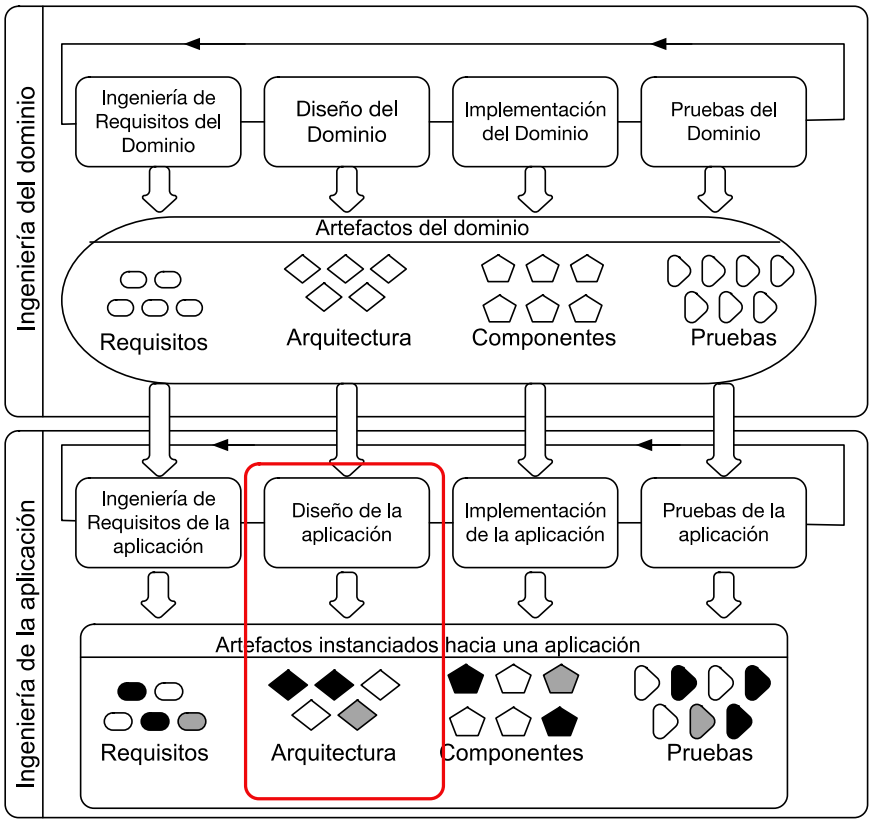

Figura 17: Etapa del Diseño de la aplicación y sus artefactos

Se definen a continuación los artefactos de la aplicación correspondientes a la etapa resaltada en la figura 17, es decir, la etapa de Diseño de la aplicación.

Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

- Arquitectura del producto: Este artefacto consta de la elección de elementos de la arquitectura de referencia que serán instanciados para la creación de un producto en particular [23, 31].

    - Estructura del producto: Se seleccionan ciertos componentes de la Estructura definida en la etapa de Diseño del dominio para ser instanciados a un producto en particular.
    - Comportamiento del producto: Cada componente de la Estructura del producto tendrá un comportamiento definido previamente en la etapa de Diseño del dominio.
- Arquitectura del producto a medida: Este artefacto representa la Arquitectura para un producto en particular considerando los aspectos relacionados con los requisitos específicos de la aplicación que no fueron contemplados por la etapa de diseño del dominio [21].
    - Estructura del producto a medida: Cada Arquitectura del producto a medida tendrá una estructura asociada, la cual define tanto los componentes como las relaciones que componen la Estructura del producto a medida [29].
    - Comportamiento del producto a medida: Este artefacto define el comportamiento que tendrán los componentes de la Estructura del producto a medida [29].
- Decisiones de Diseño del producto: Este artefacto consta de las decisiones que se tomaron para diseñar la Arquitectura del Producto a medida[1], las cuales serán documentadas para el uso que se estime conveniente [29].

4.2.3. Etapa de la Implementación de la aplicación

El objetivo principal de esta etapa es proporcionar el diseño detallado y la implementación de activos de software reutilizables, basados en la arquitectura de referencia [20].

[1] Pedro Rossel. Conversación Personal. Arquitectura de LPS.

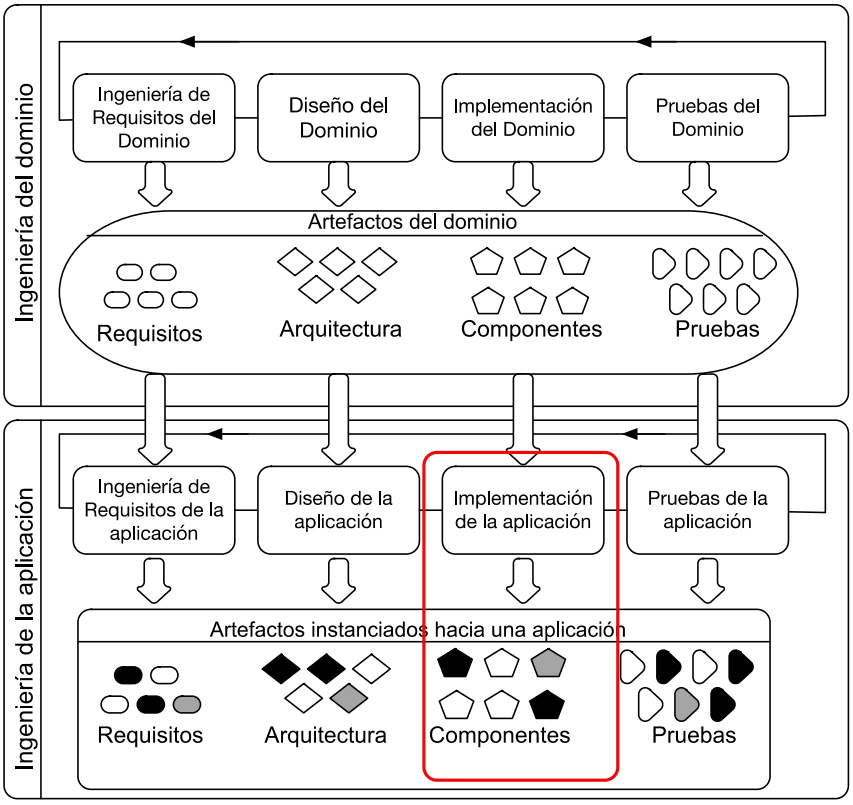

Figura 18: Etapa de Implementación de la aplicación y sus artefactos

Se definen a continuación los artefactos de la aplicación correspondientes a la etapa resaltada en la figura 18, es decir, la etapa de Implementación de la aplicación.

- Componentes reutilizables del producto: Este artefacto consta de los componentes reutilizables creados en la etapa de Implementación del dominio, tales como el código fuente y las bibliotecas, los cuales son instanciados a la etapa de implementación de la aplicación [21].
- Componentes específicos: No siempre se abarcan todas las características deseadas para una aplicación en particular, por lo tanto, a veces es necesario mejorar la funcionalidad de una aplicación o simplemente crear una nueva funcionalidad específica para la aplicación en particular [21, 36].

- Documentación del producto: Este artefacto contiene todas las decisiones que se tomaron para crear el Producto en particular, además de los Componentes reutilizables instanciados a un producto y los Componentes específicos que se crearon para el Producto en particular [20].

4.2.4. Etapa de Pruebas de aplicación

Este proceso comprende las actividades necesarias para validar y verificar una aplicación en contra de su especificación [20]. Las entradas de esta etapa son todas las aplicaciones implementadas, además de los artefactos de prueba reutilizables que se crearon en la etapa de Pruebas de Dominio. La salida de esta etapa comprende un informe de prueba con los resultados de todas las pruebas que se han realizado. Adicionalmente los defectos detectados se documentan con más detalle en los informes de problemas [20].

A continuación, se definen los artefactos para la etapa de Implementación de la aplicación. Un artefacto puede tener uno o más sub-artefactos y cada sub-artefacto puede tener uno o más sub-artefactos.

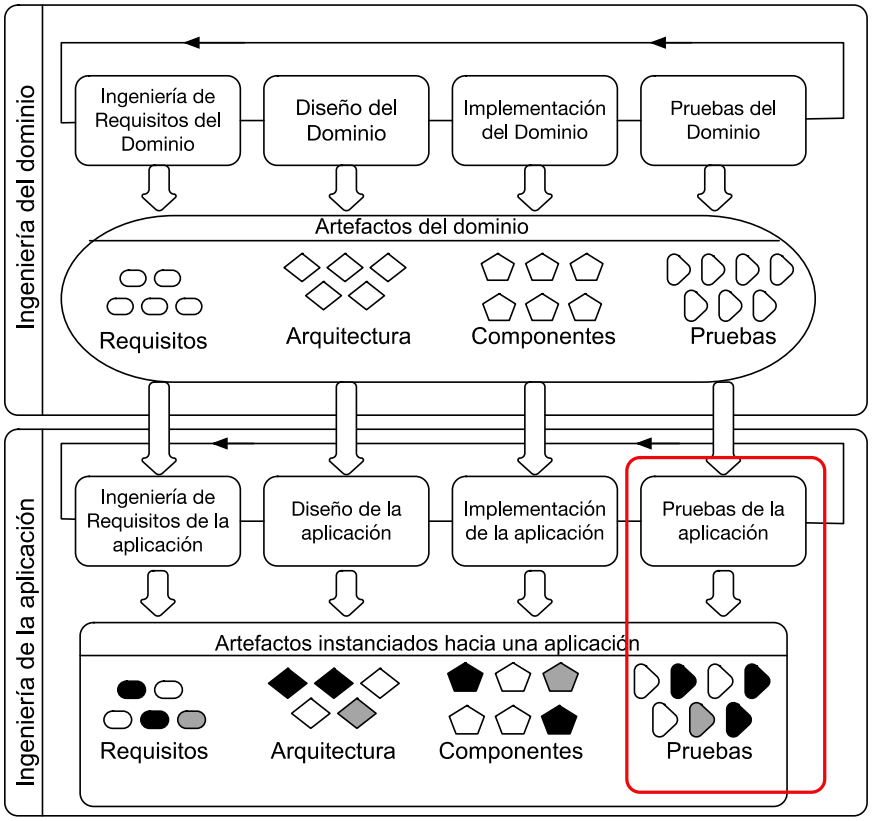

Figura 19: Etapa de Pruebas de la aplicación y sus artefactos

Se definen a continuación los artefactos de la aplicación correspondientes a la etapa resaltada en la figura 19, es decir, la etapa de Pruebas de la aplicación.

- Informe de problemas: Este artefacto comprende todos los defectos detectados en las aplicaciones implementadas por la etapa de Implementación de la aplicación [20].
- Artefactos de testeo para integración: En la etapa de Pruebas de aplicación se pueden crear artefactos de pruebas que pueden ser usadas por más de una aplicación, por lo que se evaluará si se considera como artefacto del dominio [30].
- Informe de artefactos de testeo: Este artefacto comprende todos los defectos encontrados en los artefactos de testeo provenientes de la etapa de Pruebas de

dominio, con la finalidad de obtener resultados óptimos en las pruebas de aplicación [20].

Ya definidos los artefactos para cada una de las etapas de la Ingeniería del dominio e Ingeniería de la aplicación, se procederá a explicar la trazabilidad que conectará a todos los artefactos en el modelo de trazabilidad.

La trazabilidad está conectada con la configuración y el control de versiones [28], es decir, se necesita un seguimiento de los activos cuando se desea evolucionar la LPS para saber qué activo está involucrado con qué etapa, qué artefactos se ven afectados por otros artefactos y quiénes son los encargados de realizar dicho mantenimiento.

Los escenarios de evolución que se utilizarán para la trazabilidad de artefactos en las etapas de la línea de productos de software serán tres: inserción, eliminación y modificación, definido con su sigla en inglés CUD.

En este proyecto se detallan tres trazabilidades, la interna (entre artefactos de una misma etapa), la horizontal (entre artefactos de diferentes etapas de una misma fase) y la vertical (entre artefactos de diferentes fases).

5. Trazabilidad interna

La trazabilidad interna se refiere a la relación que existe entre los artefactos de una misma etapa, esto es, tanto para las etapas de la fase de Ingeniería del dominio como para, las etapas de la fase de Ingeniería de la aplicación.

5.1. Trazabilidad interna en la fase de Ingeniería del dominio

5.1.1. Trazabilidad interna en la etapa de Ingeniería de requisitos del dominio

5.1.1.1. Ingeniería de requisitos del dominio para el escenario de evolución inserción

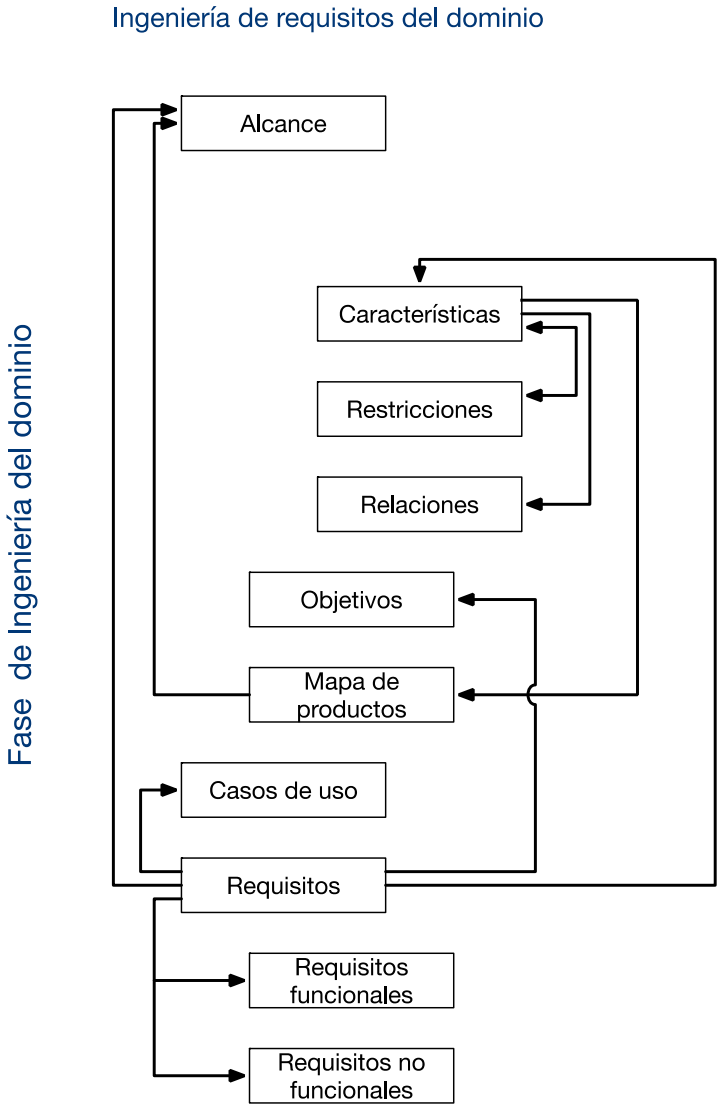

Figura 20: Trazabilidad interna en la etapa de Ingeniería de requisitos del dominio

En la figura 20 se observa la trazabilidad interna dentro de la etapa de Ingeniería de requisitos del dominio, representada por las flechas de color negro. A continuación, se

explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de la Ingeniería del dominio.

Si se agrega una *Característica* al modelo de características se deberá verificar**:**

- Mapa de productos: al insertar una nueva característica en el modelo de características, se deberá actualizar el mapa de productos, considerando que la nueva característica es para ser usada por los productos de la LPS [21].
- Relaciones: se deberán agregar las relaciones correspondientes a la nueva característica en el modelo de características. Estas relaciones pueden ser entre las características y restricciones [23, 26].
- Restricciones: La nueva característica puede estar asociada a una restricción dentro del modelo de características, por lo que debe analizarse si la nueva característica posee restricciones [23, 26].

Si se agrega una *restricción* en el modelo de características se deberá verificar**:**

- Característica: las restricciones pueden aparecer a través del tiempo, por lo tanto, se deberá analizar qué características están asociadas las nuevas restricciones [2].

Si se desea ampliar el *mapa de productos,* se deberá verificar:

- Alcance: debido a que en el alcance se define en un principio los productos que se desarrollarán en la LPS [21], se debe verificar que se siga satisfaciendo el Alcance de la LPS; en caso contrario, se deberá analizar si conviene en términos de esfuerzo realizar la modificación del alcance de la LPS.

Si se desea agregar un *requisito* se deberá verificar:

- Características: se deberá analizar si el nuevo requisito será considerado como una característica dentro del modelo de características [20].

---

[2] Pedro Rossel. Conversación personal. Restricciones del modelo de características.

- Alcance: ante una inserción de un requisito se debe corroborar que éste satisface al Alcance de la LPS [21].
- Caso de Uso: se deberá actualizar el caso de uso usando el o los requisitos nuevos que se incluyan en el sistema para dar a conocer el comportamiento del sistema con los nuevos requisitos [31].
- Requisito funcional: Se deberá agregar el requisito funcional, es decir, la acción que realizará el requisito [3].
- Requisito no funcional: se deberá agregar el requisito no funcional, es decir, cómo se realizará la acción del requisito. Por ejemplo, si la realizará en un tiempo determinado [4].
- Objetivos: se deberá analizar si la inserción del nuevo requisito modificará los objetivos definidos previamente, ya que el objetivo define lo que el sistema en consideración debe alcanzar [20].

[3] Pedro Rossel. Conversación personal. Requisitos funcionales del sistema.

[4] Pedro Rossel. Conversación personal. Requisitos no funcionales del sistema.

5.1.1.2. Ingeniería de requisitos del dominio para el escenario de evolución de modificación

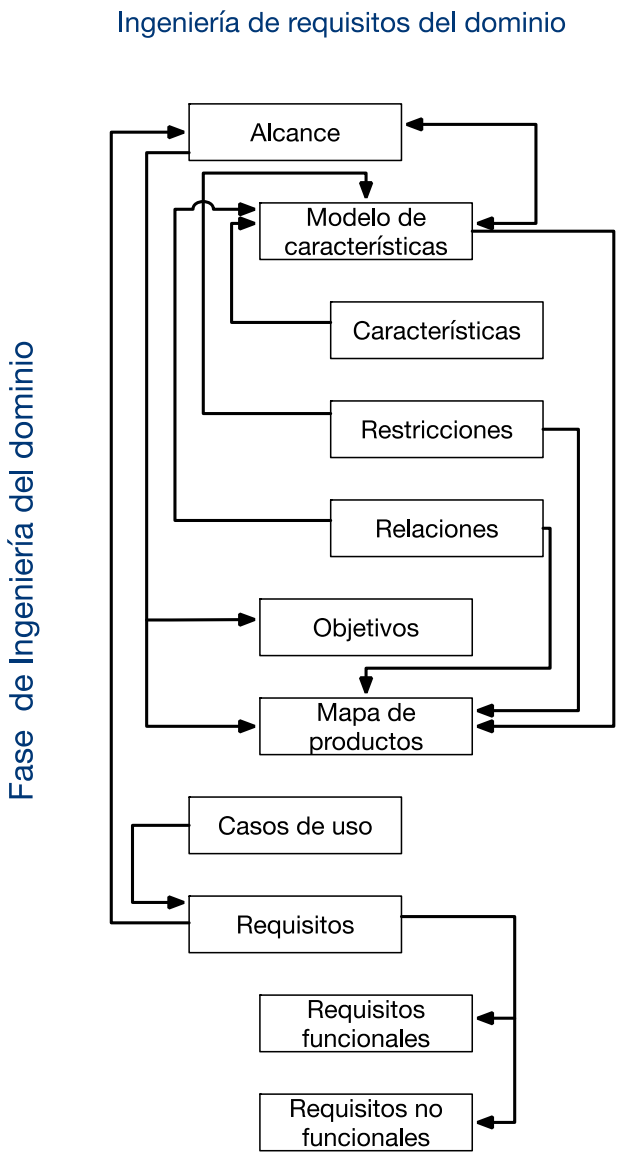

Figura 21: Trazabilidad interna en la etapa de Ingeniería de requisitos del dominio, escenario de evolución modificación

En la figura 21 se observa la trazabilidad interna dentro de la etapa de Ingeniería de requisitos del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de la Ingeniería del dominio.

Ante una modificación del *alcance* de la LPS se ven afectados:

- Modelo de Características: ya que el modelo de características es parte del alcance [21], se debe verificar ante cualquier actualización que el modelo de características sigue satisfaciendo al alcance de la LPS; en caso contrario, se deberá realizar modificaciones al modelo de características.
- Objetivos: ya que los objetivos son parte del alcance [21], se debe verificar ante cualquier modificación del alcance si es que el objetivo es el mismo definido previamente; en caso contrario, se deberá analizar si conviene o no modificar el o los objetivos.
- Mapa de Productos: ya que el mapa de productos es parte del alcance [21], se debe verificar ante cualquier modificación del alcance, que el mapa de productos tenga todos los productos y características que se requieren para satisfacer el alcance modificado.

Ante una modificación en los *requisitos* se deberá analizar**:**

- Alcance: ante cualquier modificación de los requisitos hay que verificar si es que siguen satisfaciendo al alcance definido previamente [21]; en caso contrario, se deberá analizar si conviene realizar el cambio de algún requisito.
- Requisitos funcionales: al modificar un requisito, se debe analizar si es un cambio en la parte funcional del requisito, es decir, si es que dicho requisito continúa o no realizando la misma acción.
- Requisitos no funcionales: al modificar un requisito, se debe analizar si lo que se desea modificar es la característica de funcionamiento del requisito, por ejemplo, si realizará la acción en la misma cantidad de tiempo o rapidez, etc.

Si se desea modificar un *caso de uso* se deberá analizar**:**

- Requisitos: si se desea modificar algún caso de uso, de deberá determinar si se sigue cumpliendo el escenario descrito por el caso de uso para con los requisitos. Por ejemplo, si se desea modificar el comportamiento del sistema, se deberá analizar si con el mismo requisito se puede seguir satisfaciendo el comportamiento deseado del sistema [31].

Si se desea modificar el *modelo de características* se deberá analizar**:**

- Alcance: si se desea realizar alguna modificación del modelo de características, se debe verificar la satisfacción del alcance; en caso contrario, se deberá analizar si conviene o no realizar el cambio, debido al esfuerzo mayor que podría ocasionar modificar el Alcance de la LPS.
- Mapa de productos: al realizar alguna modificación del modelo de características, se deberá verificar si existe implicancia en el mapa de productos, por ejemplo, la eliminación o inserción de alguna característica en el modelo de características, podría implicar la eliminación o inserción de un nuevo producto en el mapa de productos [21].

Si se desea modificar una *característica* se deberá analizar**:**

- Modelo de características: debido a que las características son parte del modelo de características, una modificación en éstas, podría ocasionar una modificación en el modelo de características (se refiere a los componentes que la componen, como las restricciones y relaciones).

Si se desea modificar una *restricción* se deberá analizar:

- Modelo de características: debido a que las restricciones son parte del modelo de características, alguna modificación en éstas podría ocasionar una modificación en el modelo de características.
- Mapa de productos: a partir de las restricciones se definen los distintos productos del mapa de productos [21], por ende una modificación en éstas podría ocasionar un cambio en el mapa de productos.

Si se desea modificar una *relación* se deberá analizar**:**

- Modelo de características: debido a que las relaciones son el vínculo entre las características y las restricciones [21], habrá que verificar ante una modificación de éstas, si es que el modelo de características no se modifica.
- Mapa de productos: debido a que las relaciones son el vínculo entre las características y los productos [21], se debe verificar si es que al modificar una relación el mapa de productos ha cambiado.

5.1.1.3. Ingeniería de requisitos del dominio, para el escenario de evolución eliminación

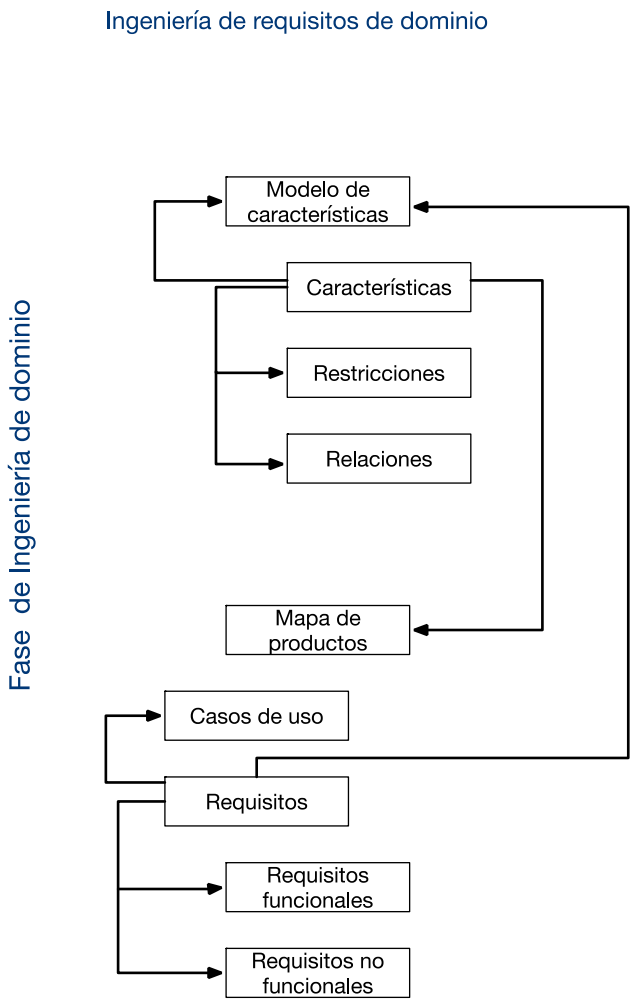

Figura 22: Trazabilidad interna en la etapa de Ingeniería de requisitos del dominio, para el escenario de evolución de eliminación

En la figura 22 se observa la trazabilidad interna dentro de la etapa de Ingeniería de requisitos del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de la Ingeniería del dominio.

Ante la eliminación de una *característica* se deberá analizar [20]:

- Modelo de Características: se deberá actualizar el modelo de características quitando la característica que ya no formará parte de éste.
- Restricción: si la característica poseía restricción dentro del modelo de características, ésta también debe ser eliminada.
- Relación: las relaciones que estaban ligadas a la característica eliminada también debe ser eliminada del modelo de características.
- Mapa de Productos: el mapa de productos posee todos los productos con sus respectivas características, por lo que también debe ser actualizado.

Ante la eliminación de un *requisito* se deberá analizar [20]:

- Modelo de Características: hay requisitos que a su vez son características dentro del sistema, por lo que debe analizarse el modelo de características en conjunto con los elementos que la componen (restricciones y relaciones).
- Caso de Uso: se deberá actualizar el o los casos de uso que contengan algún requisito eliminado del sistema.
- Requisito funcional: se deberá eliminar la parte funcional del requisito eliminado, es decir, la actividad que el sistema hacía, la cual dependía del requisito eliminado.
- Requisito no funcional: se deberán eliminar los requisitos funcionales que eran parte del requisito eliminado.

5.1.2. Trazabilidad interna en la etapa de Diseño del dominio

5.1.2.1. Diseño del dominio, para el escenario de evolución inserción

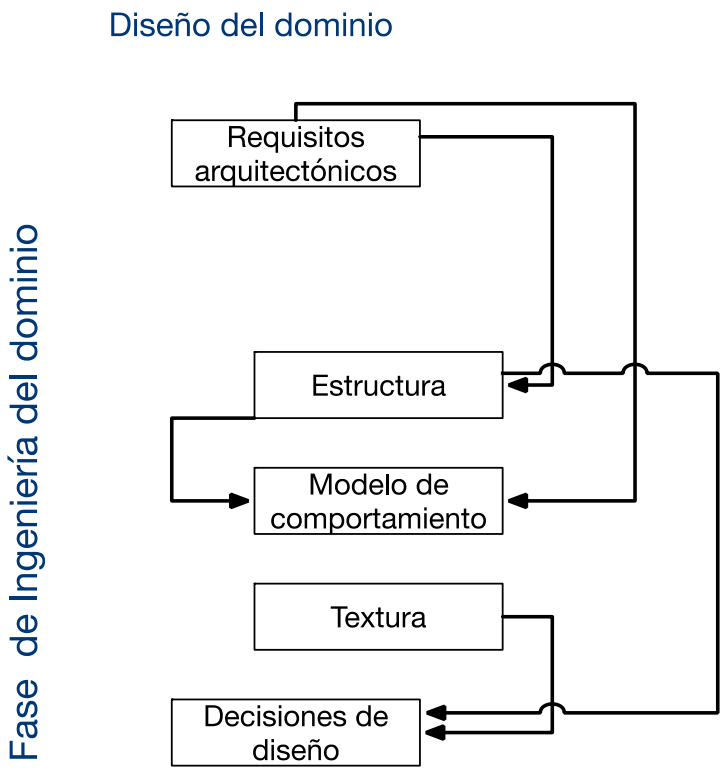

Figura 23: Trazabilidad interna en la etapa de Diseño del dominio para el escenario de evolución de inserción

En la figura 23 se observa la trazabilidad interna dentro de la etapa de Diseño del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Diseño del dominio.

Ante una inserción de un *Requisito Arquitectónico* se deberá analizar**:**

- Estructura: los requisitos arquitectónicos darán pie al desarrollo de la arquitectura de referencia [24], específicamente a la parte estructural de ésta, por lo que ante una inserción de un requisito arquitectónico, se deberá analizar si cambia la estructura de la arquitectura de referencia.
- Modelo de comportamiento: ya que se establece el comportamiento del sistema en base a los requisitos diseñados, se deberá actualizar el modelo de comportamiento ante una inserción de un requisito [24].

Ante la ampliación de la *textura* se debe analizar:

- Decisiones de diseño: deben ser documentadas todas las decisiones de diseño, ya sea si se agregó un patrón de diseño diferente o algún estilo arquitectónico [24].

Ante la ampliación de la *estructura* se deberá analizar:

- Modelo de comportamiento: se deberá agregar en el modelo de comportamiento, el comportamiento correspondiente al nuevo componente de la Estructura [21].
- Decisiones de diseño: se deberán documentar todos los componentes de la estructura, incluyendo la variabilidad en el diseño [20], por lo que deberá actualizarse el documento de las decisiones de diseño.

5.1.2.2. Diseño del dominio, para el escenario de evolución modificación

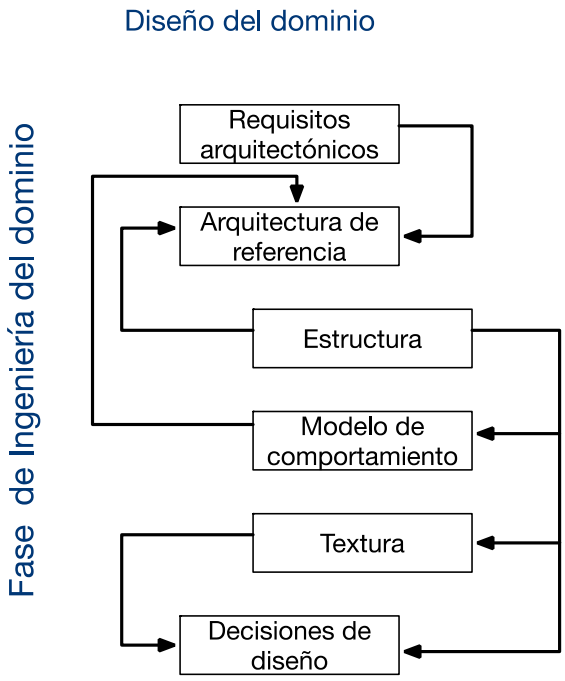

Figura 24: Trazabilidad interna en la etapa de Diseño del dominio, escenario de evolución de modificación

En la figura 24 se observa la trazabilidad interna dentro de la etapa de Diseño del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Diseño del dominio.

Ante una modificación en los *requisitos arquitectónicos* se deberá analizar**:**

- Arquitectura de Referencia: ya que son los requisitos en evolución, y los identificados como más relevantes, darán pie al desarrollo de la arquitectura de referencia, por lo que cualquier modificación de éstos, repercutirá en la arquitectura de referencia [24].

Ante una modificación en la *estructura* se deberá analizar**:**

- Arquitectura de Referencia**:** debido a que la variabilidad de la arquitectura se incorpora principalmente en la estructura [20], deberá ser actualizada la Arquitectura de Referencia.
- Textura: la estructura define qué partes se construyen por separado, mientras que la textura determina las reglas generales que cada una de las partes tiene que obedecer [25], por lo tanto, si se modifica la estructura, se deberá revisar la textura.
- Modelo de Comportamiento: si se modifica la estructura, variará el comportamiento de esos componentes, por lo que se deberá actualizar el modelo de comportamiento [21].
- Decisiones del diseño: si se modifica la Estructura se deberá documentar cuales fueron las decisiones que se tomaron para realizar dicho cambio, para tener en cuenta con futuros escenarios de evolución [5].

Ante una modificación en la *textura* se deberá analizar**:**

- Decisiones de diseño: ya que la textura consiste en las reglas generales de desarrollo para la realización del sistema, tales como patrones de diseño y estilos arquitectónicos [24], se deberán actualizar las decisiones de diseño que se tomaron para realizar algún tipo de modificación en la textura.

Ante una modificación del *modelo de comportamiento* se deberá analizar**:**

- Arquitectura de Referencia: como el comportamiento refleja las posibles interacciones entre los componentes detectados por la Estructura [21], cualquier modificación del modelo de comportamiento podría modificar la arquitectura de referencia.

[5] Pedro Rossel. Conversación Personal. Decisiones de diseño.

5.1.2.3. Diseño del dominio, para el escenario de evolución eliminación

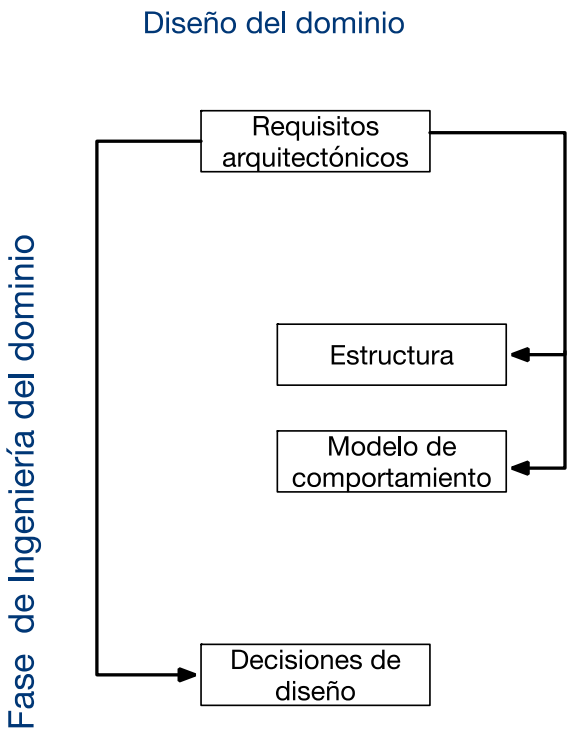

Figura 25: Trazabilidad interna en la etapa de Diseño del dominio para el escenario de evolución de eliminación

En la figura 25 se observa la trazabilidad interna dentro de la etapa de Diseño del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Diseño del dominio.

Ante una eliminación de un *requisito arquitectónico* se deberá analizar:

- Estructura: se deberá actualizar la parte estructural de la arquitectura de referencia debido a que ésta se creó en base a los requisitos arquitectónicos definidos previamente [24].

- Modelo de comportamiento: como se estableció el comportamiento del sistema en base a los requisitos arquitectónicos, se deberá actualizar el modelo de comportamiento al momento de eliminar un requisito arquitectónico [24].
- Decisiones de diseño: se deberán documentar las razones por las que se decidió eliminar algún Requisito arquitectónico, para usos futuros.

5.1.3. Trazabilidad interna en la etapa de Implementación del dominio

5.1.3.1. Implementación del dominio, para el escenario de evolución inserción

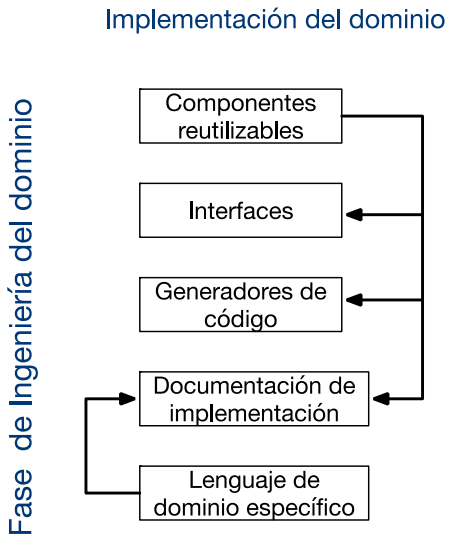

Figura 26: Trazabilidad interna en la etapa de Implementación del dominio, escenario de evolución de inserción

En la figura 26 se observa la trazabilidad interna dentro de la etapa de Implementación del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Implementación del dominio.

Ante una inserción de un *componente reutilizable* se deberá analizar:

- Interfaces: se deberá crear la parte externamente visible del nuevo componente, además de conectar el nuevo componente con los otros ya creados previamente [20].
- Generadores de Código: se deberá actualizar el o los generadores de código que permita integrar el nuevo componente reutilizable [21].
- Documentación de implementación: se deberá actualizar la información contenida en la documentación de implementación, ya que esta debe contener todos los componentes reutilizables construidos en la etapa de Implementación del Dominio [20].

Si se desea agregar un *lenguaje de dominio específico* se deberá revisar:

- Documentación de implementación: debido a que esta debe contener además de la información relacionada a los componentes reutilizables del producto, el lenguaje de dominio específico [20].

5.1.3.2. Implementación del dominio, para el escenario de evolución modificación

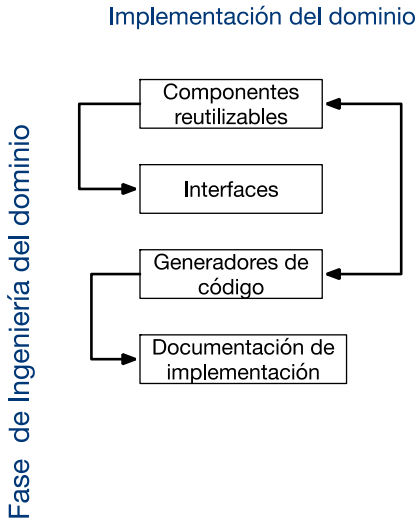

Figura 27: Trazabilidad interna en la etapa de Implementación del dominio, escenario de evolución de modificación.

En la figura 27 se observa la trazabilidad interna dentro de la etapa de Implementación del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Implementación del dominio.

Ante una modificación de los *componentes reutilizables* se deberá analizar**:**

- Interfaces: debido a que las interfaces son las partes externamente visibles de los componentes, además de ser el medio para conectar los componentes, podrían verse afectadas ante una modificación de los componentes [20].
- Generadores de Código: ante cualquier modificación en los componentes reutilizables se debe verificar que el generador de código siga satisfaciendo todos los componentes reutilizables [21].

Ante una modificación a los *generadores de código* se deberá analizar**:**

- Componentes Reutilizables: cuando se realiza alguna modificación en los generadores de código, se debe verificar que se puedan incorporar todos los componentes reutilizables para la posterior integración de éstos [21].
- Documentación de Implementación: todas las modificaciones que se realicen a los componentes reutilizables deben ser documentadas para una posterior revisión en casos de realizar actualizaciones en el sistema [20].

5.1.3.3. Implementación del dominio, para el escenario de evolución eliminación

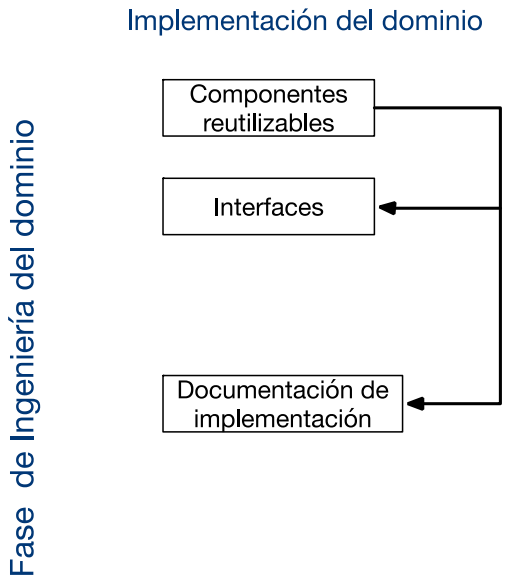

Figura 28: Trazabilidad interna en la etapa de Implementación del dominio, escenario de evolución de eliminación

En la figura 28 se observa la trazabilidad interna dentro de la etapa de Implementación del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Implementación del dominio.

En el caso de eliminar un *componente reutilizable* se deberá revisar:

- Interfaces: se deberá eliminar la interfaz asociada al componente reutilizable eliminado [20].
- Generadores de Código: se deberá actualizar el generador de código, debido a que ya no deberá contar con la opción de elegir alguna función asociada al componente que se eliminó.

- Documentación de implementación: se deberá actualizar la información asociada a los componentes reutilizables, eliminando aquella relacionada con el componente eliminado [20].

5.1.4. Trazabilidad interna en la etapa de Pruebas del dominio

5.1.4.1. Pruebas del dominio, para el escenario de evolución inserción

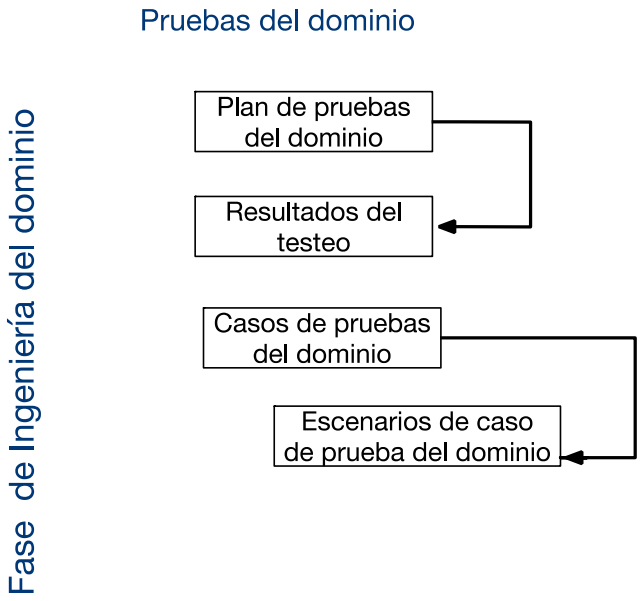

Figura 29: Trazabilidad interna en la etapa de Pruebas del dominio, para el escenario de evolución de inserción

En la figura 29 se observa la trazabilidad interna dentro de la etapa de Pruebas del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Pruebas del dominio.

Al agregar un *caso de prueba del dominio* se deberá analizar:

- Escenario de prueba del dominio: si se agregan casos de prueba del dominio se deben agregar a su vez los escenarios de caso de prueba del dominio, los cuales describen una secuencia específica de acciones a ejecutar, esto se debe a que cada caso de prueba del dominio incluye a su vez uno o varios escenarios de casos de prueba [27].

Al agregar un *plan de pruebas del dominio* se deben analizar:

- Resultados del testeo: ya que estos comprenden además de los resultados de las pruebas realizadas en las pruebas del dominio, el resultado de las pruebas realizadas en los artefactos reutilizables, los cuales son proporcionados por el plan de pruebas del dominio [20].

5.1.4.2. Pruebas del dominio, para el escenario de evolución modificación

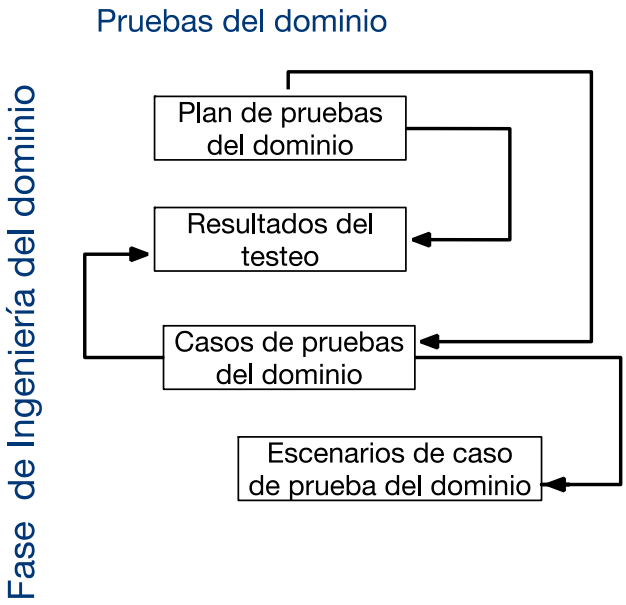

Figura 30: Trazabilidad interna en la etapa de Pruebas del dominio para el escenario de evolución de modificación

En la figura 30 se observa la trazabilidad interna dentro de la etapa de Pruebas del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Pruebas del dominio.

Ante una modificación en el *plan de pruebas del dominio* se deberá analizar**:**

- Casos de prueba del dominio: se debe verificar si los casos de pruebas del dominio se mantienen o varían, debido a que estos son definidos por el plan de pruebas [20].
- Resultados del testeo: una modificación en el plan de pruebas del dominio puede hacer variar los resultados del testeo, debido a que éstos en parte muestran los resultados de las pruebas que se realizan a los artefactos de prueba reutilizables, los cuales fueron definidos por el plan de pruebas del dominio [20].

Ante una modificación en los *casos de pruebas del dominio* se deberá analizar:

- Resultados del testeo: en los casos de pruebas del dominio se definen entre otras cosas los datos de entrada y de salida esperados para la prueba, y una modificación en éstos podría generar una variación en los resultados del testeo [27].
- Escenarios de Caso de Prueba del Dominio: si se modifica un caso de prueba del dominio, podría generar modificaciones en los escenarios de caso de prueba debido a que los casos de prueba son los que definen uno o varios casos de prueba a ejecutar [27].

5.1.4.3. Pruebas del dominio, para el escenario de evolución eliminación

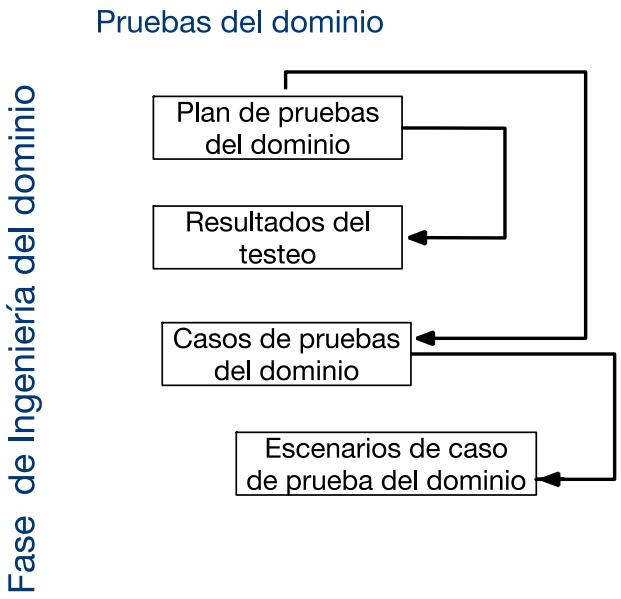

Figura 31: Trazabilidad interna en la etapa de Pruebas del dominio, para el escenario de evolución de eliminación

En la figura 31 se observa la trazabilidad interna dentro de la etapa de Pruebas del dominio, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Pruebas del dominio.

Si se desea eliminar un *caso de prueba de dominio* se debe analizar:

- Escenario de caso de prueba de dominio: se deberá eliminar el o los escenarios de caso de prueba de dominio creados por el caso de prueba, ya que estos escenarios fueron creados para cumplir con el objetivo de prueba dentro del caso de prueba definido por éste [27].

Si se desea eliminar un *plan de pruebas del dominio* se debe analizar:

- Resultados del testeo: se deberá eliminar el resultado del testeo de las pruebas realizadas a los artefactos reutilizables creados por el plan de pruebas del dominio eliminado [20].
- Casos de prueba del dominio: se deben eliminar los casos de prueba del dominio creados por el plan de pruebas del dominio que fue eliminado, ya que estos fueron creados con el fin de satisfacer a éste último.

5.2. Trazabilidad interna en fase de Ingeniería de la aplicación

5.2.1. Trazabilidad interna en la etapa de Requisitos de la aplicación

5.2.1.1. Requisitos de la aplicación, para el escenario de evolución inserción

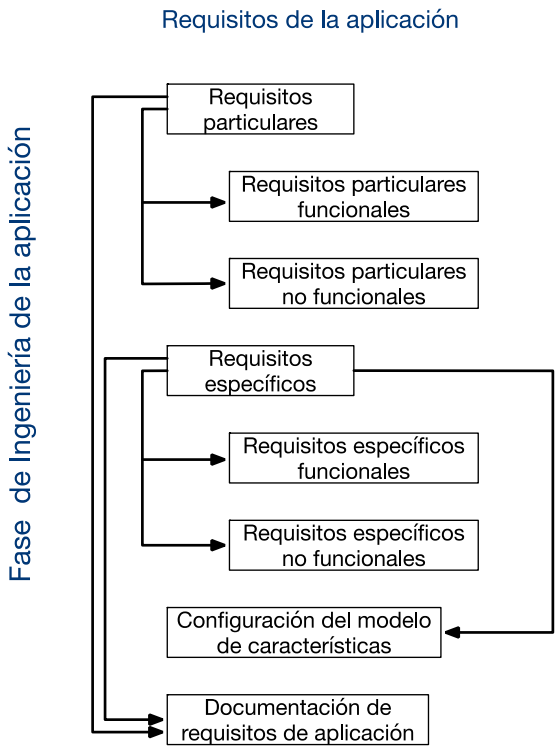

Figura 32: Trazabilidad interna en la etapa de Requisitos de la aplicación, para el escenario de evolución de inserción.

En la figura 32 se observa la trazabilidad interna dentro de la etapa de Requisitos de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Requisitos de la aplicación.

Ante une inserción de un *requisito particular* se deberá analizar:

- Requisito particular funcional: se deberá incluir el requisito funcional del requisito particular instanciado [24].
- Requisito particular no funcional: se deberá incluir el requisito no funcional o atributo de calidad que involucre el requisito particular, por ejemplo, alguna restricción del requisito particular [24].
- Documentación de requisitos de aplicación: se deben incluir en este documento todos los Requisitos particulares instanciados para cada producto de la LPS [24].

Ante una inserción de un *requisito específico* se debe analizar:

- Documentación de requisitos de la aplicación: se deben incluir en este documento todos los requisitos específicos que fueron creados y aceptados por las partes interesadas en la etapa de Ingeniería de la aplicación.
- Requisito específico funcional: al agregar un requisito específico se deberá documentar a su vez los requisitos funcionales del nuevo requisito específico.
- Requisito específico no funcional: al agregar un requisito específico se deberá documentar a su vez los requisitos no funcionales o restricciones del sistema con respecto al nuevo requisito específico agregado.
- Configuración del modelo de característica: si el nuevo requisito específico creado podría formar parte de una característica dentro del sistema, por lo que deberá ser agregado al modelo de características del producto en particular [24].

5.2.1.2. Requisitos de la aplicación, para el escenario de evolución modificación

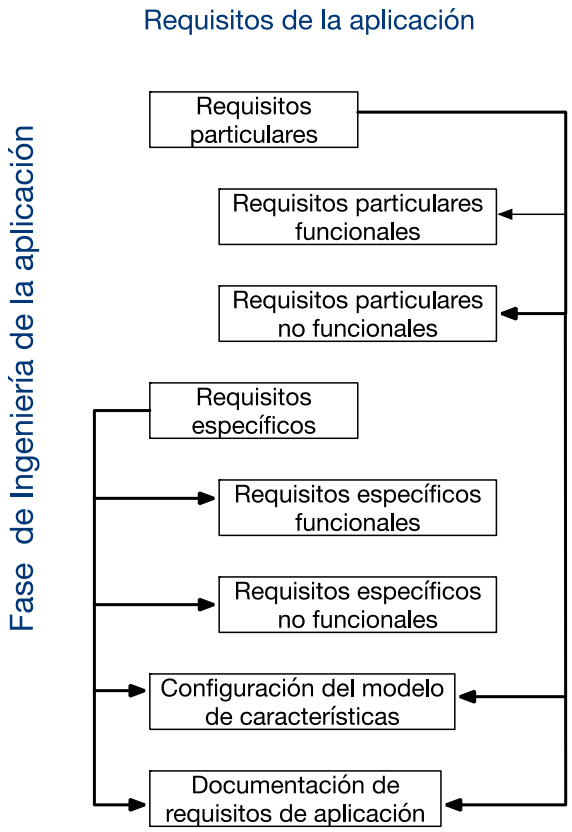

Figura 33: Trazabilidad interna en la etapa de Requisitos de la aplicación, para el escenario de evolución de modificación

En la figura 33 se observa la trazabilidad interna dentro de la etapa de Requisitos de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Requisitos de la aplicación.

Ante una modificación de los *requisitos particulares* se deberá analizar [24]:

- Requisitos particulares funcionales: ante una modificación de un requisito particular, se debe analizar si es un cambio en la parte funcional del requisito, es decir, se analizará si el requisito continuará o no realizando la misma acción.
- Requisitos particulares no funcionales: al modificar un requisito particular, se debe analizar si es un cambio en la parte no funcional del requisito, es decir, cómo hará la acción o bien si se ve involucrado algún atributo de calidad debido al cambio.
- Documentación de Requisitos de la aplicación: se deberá actualizar la documentación de requisitos de la aplicación con todos los cambios realizados a los requisitos particulares.
- Configuración del modelo de características: el requisito particular modificado podría formar parte de una característica dentro del sistema, por lo que deberá ser agregado al modelo de características del producto en particular.

Ante una modificación de un *requisito específico* se deberá analizar [24]:

- Requisitos específicos funcionales: ante una modificación de un requisito específico, se debe analizar si es un cambio en la parte funcional del requisito, es decir, se analizará si el requisito continuará o no realizando la misma acción.
- Requisitos específicos no funcionales: al modificar un requisito específico, se debe analizar si es un cambio en la parte no funcional del requisito, es decir, cómo hará la acción o bien si se ve involucrado algún atributo de calidad debido al cambio.
- Documentación de requisitos de la aplicación: se deberá actualizar la documentación de requisitos de aplicación con todos los cambios realizados a los requisitos específicos para un producto en particular.
- Configuración del modelo de características: el requisito específico modificado podría formar parte de una característica dentro del sistema, por lo que deberá ser agregado al modelo de características del producto en particular.

5.2.1.3. Requisitos de la aplicación, para el escenario de evolución eliminación

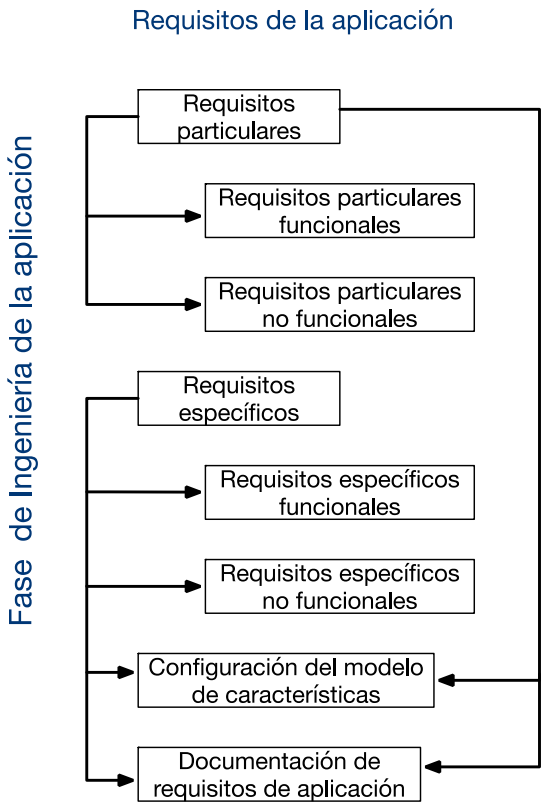

Figura 34: Trazabilidad interna en la etapa de Requisitos de la aplicación, para el escenario de evolución de eliminación

En la figura 34 se observa la trazabilidad interna dentro de la etapa de Requisitos de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Requisitos de la aplicación.

Al eliminar un *requisito particular* se deberá revisar [24]:

- Requisito particular funcional: se deberá eliminar la parte funcional del requisito particular eliminado.
- Requisito particular no funcional: se deberá eliminar la parte no funcional del requisito particular eliminado.
- Configuración del modelo de característica: si el requisito particular eliminado era considerado como característica dentro del sistema, deberá ser eliminado de la Configuración del modelo de características.
- Documentación de Requisitos de aplicación: se deberá actualizar la Documentación de requisitos de aplicación con los Requisitos particulares vigentes.

Al eliminar un *requisito específico* se deberá revisar [24]:

- Requisito específico funcional: por consecuencia, se deberá eliminar la parte funcional del Requisito funcional eliminado.
- Requisito particular no funcional: por consecuencia, se deberá eliminar la parte no funcional del Requisito específico eliminado.
- Configuración del modelo de característica: si el requisito específico eliminado era considerado como característica dentro del sistema, deberá ser eliminado del modelo de características para un producto en particular.
- Documentación de requisitos de aplicación: se deberá actualizar la documentación de requisitos de aplicación con los requisitos específicos vigentes.

5.2.2. Trazabilidad interna en la etapa de Diseño de la aplicación

5.2.2.1. Diseño de la aplicación, para el escenario de evolución inserción

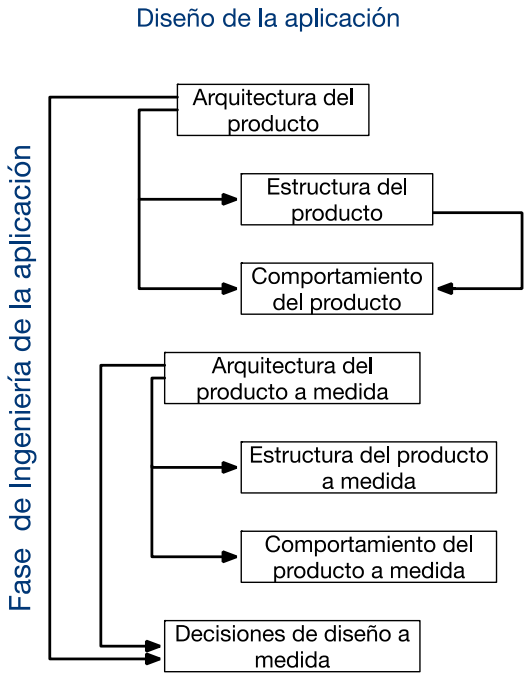

Figura 35: Trazabilidad interna en la etapa de Diseño de la aplicación, para el escenario de evolución de inserción

En la figura 35 se observa la trazabilidad interna dentro de la etapa de Diseño de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Diseño de la aplicación.

Si se desea agregar un elemento a la *arquitectura del producto* se deberá analizar [29]:

- Estructura del producto: se deberá agregar el nuevo componente de la estructura que ha sido instanciado a la arquitectura del producto.
- Comportamiento del producto: se deberá agregar el comportamiento al modelo de comportamiento del producto particular, asociado al nuevo elemento de la estructura.
- Decisiones de diseño a medida: se deberá documentar las decisiones que se tomaron para la inserción del nuevo elemento a la Arquitectura del producto particular.

Si se desea agregar un elemento a la *arquitectura del producto a medida* se deberá analizar [29]:

- Estructura del producto a medida: debido a que la arquitectura del producto a medida consta de una parte estructural, se deberán agregar todos los elementos que componen la estructura de la arquitectura del producto a medida que se desea agregar.
- Comportamiento del producto a medida: debido a que la Arquitectura del producto a medida consta de una vista de comportamiento, se deberán agregar todos los elementos que componen esta vista para la arquitectura del producto a medida que se desea agregar.
- Decisiones de diseño: se deberán documentar las decisiones que se tomaron para realizar la inserción de algún elemento en la Arquitectura del producto a medida.

5.2.2.2. Diseño de la aplicación, para el escenario de evolución modificación

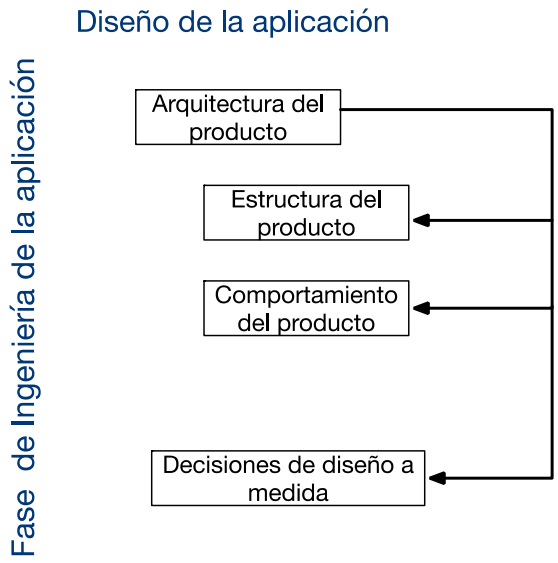

Figura 36: Trazabilidad interna en la etapa de Diseño de la aplicación, para el escenario de evolución de modificación

En la figura 36 se observa la trazabilidad interna dentro de la etapa de Diseño de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Diseño de la aplicación.

Ante una modificación de la *arquitectura del producto* se deberá analizar:

- Estructura del producto: se debe actualizar la parte estructural de la arquitectura del producto, analizando el esfuerzo requerido para realizar dichos cambios [20].
- Comportamiento del producto: se debe actualizar el comportamiento que tendrán los elementos de la arquitectura del producto modificada [24].
- Decisiones de diseño del producto: se deberán documentar las decisiones que se tomaron para realizar la modificación de la arquitectura del producto [20].

5.2.2.3. Diseño de la aplicación, para el escenario de evolución eliminación

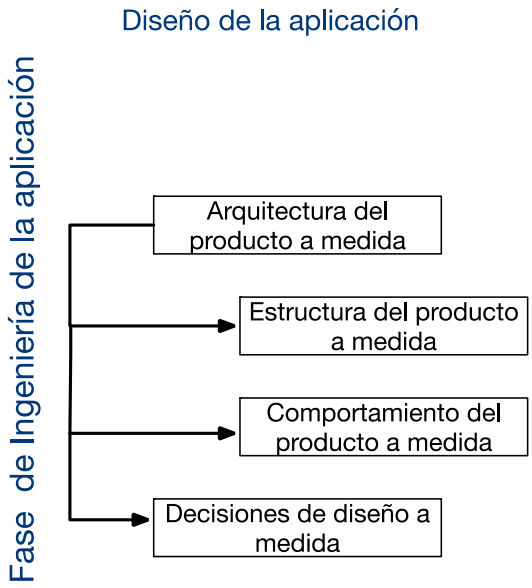

Figura 37: Trazabilidad interna en la etapa de Diseño de la aplicación, para el escenario de evolución de eliminación

En la figura 37 se observa la trazabilidad interna dentro de la etapa de Diseño de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Diseño de la aplicación.

Ante una eliminación de la *arquitectura del producto a medida* se debe analizar [29]:

- Estructura del producto a medida: se deberá eliminar la estructura del producto a medida, debido a que es parte de la arquitectura eliminada.
- Comportamiento del producto a medida: se deberá eliminar el comportamiento de los elementos de la estructura debido a que es parte de la Arquitectura eliminada.
- Decisiones de diseño del producto: se deberá documentar cuáles fueron las decisiones que se tomaron para eliminar la arquitectura del producto a medida, para usos futuros.

5.2.3. Trazabilidad interna en la etapa de Implementación de la aplicación

5.2.3.1. Implementación de la aplicación, para el escenario de evolución inserción

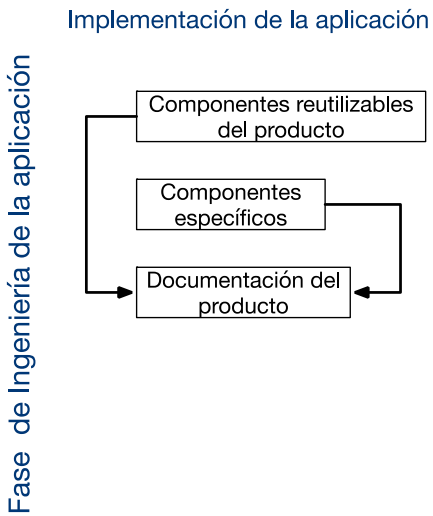

Figura 38: Trazabilidad interna en la etapa de Implementación de la aplicación, para el escenario de evolución de inserción

En la figura 38 se observa la trazabilidad interna dentro de la etapa de Implementación de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Implementación de la aplicación.

Ante la inserción de un *componente específico* se deberá analizar:

- Documentación del producto: se deberá documentar el código fuente o las bibliotecas que han sido añadidas para la implementación del producto en particular [20].

Ante la inserción de un *componente reutilizable del producto* se debe analizar:

- Documentación del producto: se deberá documentar cuáles fueron los módulos de código fuente o bibliotecas que han sido instanciados para implementar un producto en particular [20].

5.2.3.2. Implementación de la aplicación, para el escenario de evolución modificación

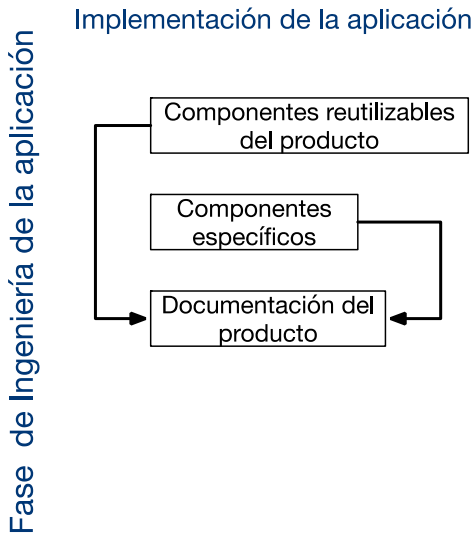

Figura 39: Trazabilidad interna en la etapa de Implementación de la aplicación, para el escenario de evolución de modificación

En la figura 39 se observa la trazabilidad interna dentro de la etapa de Implementación de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Implementación de la aplicación.

Ante la modificación de un *componente reutilizable del producto* se debe analizar:

- Documentación del producto: se deberá actualizar la documentación del producto con todos los cambios realizados al componente reutilizable [24].

Ante la modificación de un *componente específico* se debe analizar:

- Documentación del producto: se deberá actualizar la documentación del producto con todas las modificaciones que se realizaron al componente específico del producto, ya sea modificaciones de código fuente o bibliotecas utilizadas [24].

5.2.3.3. Implementación de la aplicación, para el escenario de evolución eliminación

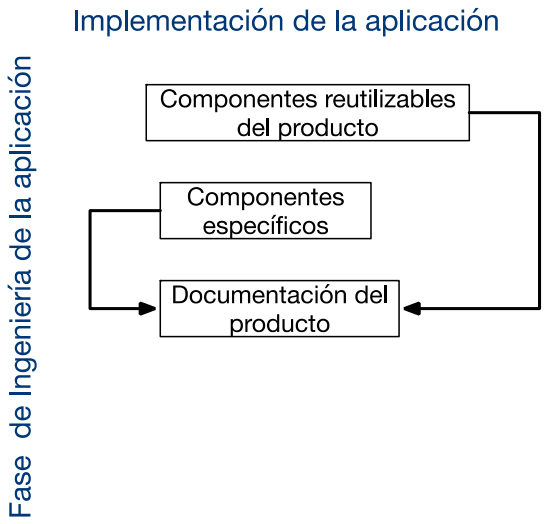

Figura 40: Trazabilidad interna en la etapa de Implementación de la aplicación, para el escenario de evolución de eliminación

En la figura 40 se observa la trazabilidad interna dentro de la etapa de Implementación de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Implementación de la aplicación.

Ante la eliminación de un *componente reutilizable del producto* se deberá analizar:

- Documentación del producto: se deberá actualizar la documentación del producto justificando la eliminación de un componente reutilizable para el producto en particular [24].

Ante la eliminación de un *componente específico* se deberá analizar:

- Documentación del producto: se deberá actualizar la documentación del producto justificando la eliminación de un componente específico para el producto en particular [24].

5.2.4. Trazabilidad interna en la etapa de Pruebas de la aplicación

5.2.4.1. Pruebas de la aplicación, para el escenario de evolución inserción

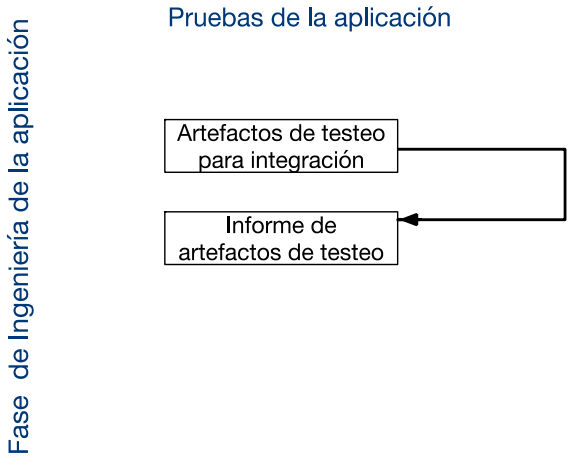

Figura 41: Trazabilidad interna en la etapa de Pruebas de la aplicación, para el escenario de evolución de inserción

En la figura 41 se observa la trazabilidad interna dentro de la etapa de Pruebas de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de la etapa de Pruebas de la aplicación.

Ante una inserción de un *artefacto de testeo para integración* se deberá revisar:

- Informe de artefactos de testeo: al agregar un nuevo artefacto de testeo para integración, este debe ser probado y validado por las partes interesadas para posteriormente ser añadido como artefacto reutilizable dentro de la etapa de pruebas de dominio [20], el cual además debe ser documentado en el informe de artefactos de testeo para la posterior revisión.

5.2.4.2. Pruebas de la aplicación, para el escenario de evolución modificación

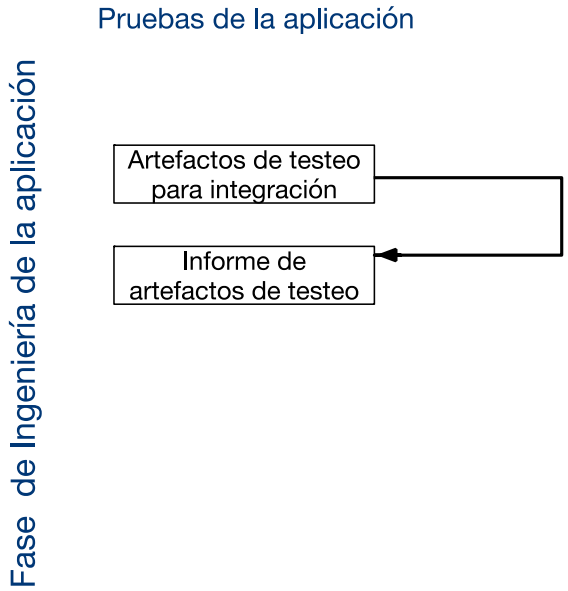

Figura 42: Trazabilidad interna en la etapa de Pruebas de la aplicación, para el escenario de evolución de modificación

En la figura 42 se observa la trazabilidad interna dentro de la etapa de Pruebas de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de la etapa de Pruebas de la aplicación.

Ante una modificación en el *artefacto de testeo para integración* se deberá revisar:

- Informe de artefactos de testeo: se deberá actualizar el informe de artefactos de testeo con las modificaciones realizadas al artefacto de testeo para integración para su posterior revisión y validación del artefacto modificado [24].

5.2.4.3. Pruebas de la aplicación, para el escenario de evolución eliminación

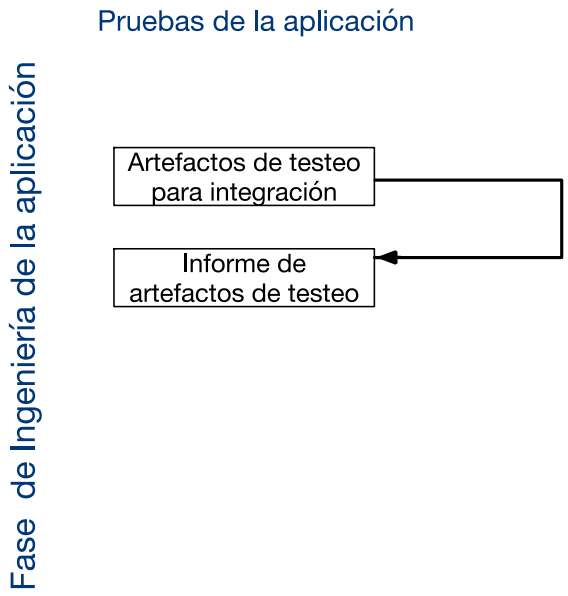

Figura 43: Trazabilidad interna en la etapa de Pruebas de la aplicación, para el escenario de evolución de eliminación

En la figura 43 se observa la trazabilidad interna dentro de la etapa de Pruebas de la aplicación, representada por las flechas de color negro. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de la etapa de Pruebas de la aplicación.

Ante una eliminación de un *artefacto de testeo para integración* se deberá revisar:

- Informe de artefactos de testeo: se deberá actualizar el informe de artefactos de testeo considerando la eliminación del artefacto, además de documentar las decisiones que se tomaron para realizar dicha eliminación [24].

5.3. Trazabilidad horizontal en fase de Ingeniería del dominio

5.3.1. Entre Ingeniería de Requisitos del Dominio y Diseño del Dominio para el escenario de evolución de inserción

Figura 44: Trazabilidad horizontal entre Ingeniería de requisitos del dominio y Diseño del dominio, para el escenario de evolución de inserción

En la figura 44 se observa la trazabilidad horizontal entre las etapas de Ingeniería de requisitos del dominio y Diseño del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante una inserción de un *requisito* se debe revisar:

- Estructura: se deberá agregar el componente de la estructura que nace a partir del nuevo requisito que será común para la LPS, debido a que la arquitectura de referencia solo satisface los requisitos comunes entre los productos de la LPS [20].
- Requisitos arquitectónicos: como en la realidad los requisitos no están listos para ser diseñados a tiempo, se debe analizar si el nuevo requisito diseñado se identifica como relevante para conformar la Arquitectura de Referencia [24]; de ser así, deberá ser agregado a la arquitectura de referencia.

5.3.2. Entre Ingeniería de Requisitos del Dominio y Diseño del Dominio para el escenario de evolución de modificación

Figura 45: Trazabilidad horizontal entre Ingeniería de requisitos del dominio y Diseño del dominio, para el escenario de evolución de modificación

En la figura 45 se observa la trazabilidad horizontal entre las etapas de Ingeniería de requisitos del dominio y Diseño del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de un *requisito* se debe analizar:

- Estructura: se debe actualizar la parte estructural de la arquitectura de referencia, considerando los cambios que se realizaron al requisito [24].
- Requisitos arquitectónicos: en caso de que el requisito que se modificó forme parte de los requisitos arquitectónicos, se deberán actualizar los requisitos arquitectónicos para luego ser diseñados por la arquitectura de referencia [24].

5.3.3. Trazabilidad horizontal entre Ingeniería de Requisitos del Dominio y Diseño del Dominio para el escenario de evolución de eliminación

Figura 46: Trazabilidad horizontal entre Ingeniería de requisitos del dominio y Diseño del dominio, para el escenario de evolución de eliminación

En la figura 46 se observa la trazabilidad horizontal entre las etapas de Ingeniería de requisitos del dominio y Diseño del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de un *requisito* se deberá analizar:

- Requisitos arquitectónicos: se deberá eliminar el requisito arquitectónico que fue diseñado a partir del requisito creado en la etapa de Ingeniería de requisitos de dominio [24].
- Estructura: se deberá actualizar la parte estructural de la arquitectura de referencia, eliminando aquel elemento que consideraba el requisito eliminado de la Ingeniería de requisitos de dominio [24].

5.3.4. Trazabilidad horizontal entre Diseño del Dominio e Implementación del Dominio para el escenario de evolución de inserción

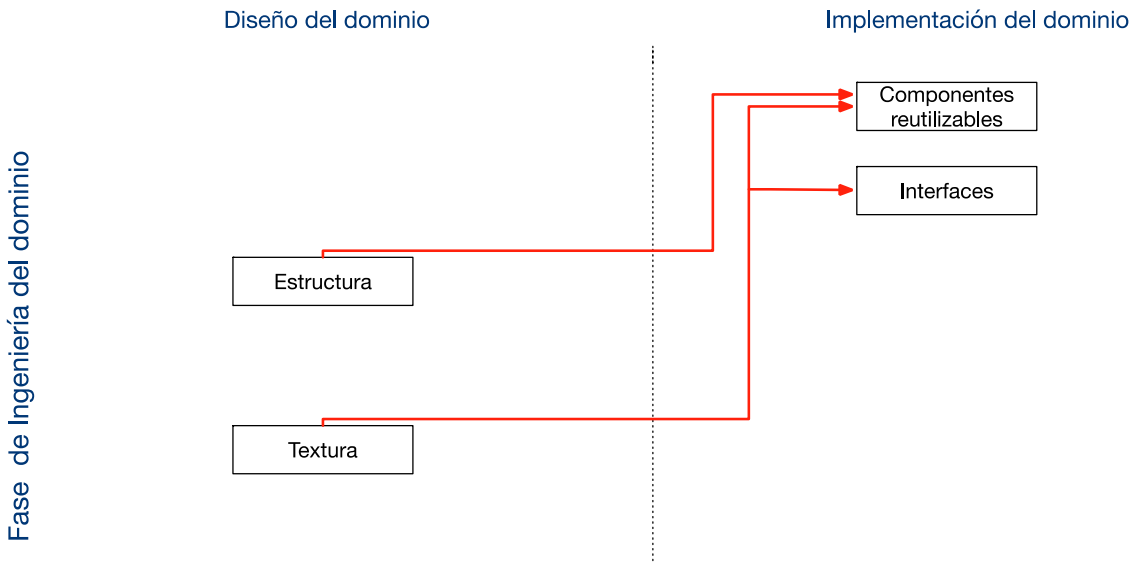

Figura 47: Trazabilidad horizontal entre Diseño del dominio e Implementación del dominio, para el escenario de evolución de inserción

En la figura 47 se observa la trazabilidad horizontal entre las etapas de Diseño del dominio e Implementación del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de una *textura* se debe analizar:

- Componentes reutilizables: debido a que pueden ser agregadas reglas de codificación y mecanismos generales para el proceso de codificación. Por lo tanto, éstas deben ser utilizadas en la implementación de los componentes reutilizables para asegurar la reusabilidad de todo el diseño y los artefactos de implementación [24].
- Interfaces: debido a que la textura provee, entre otras cosas, el diseño de interfaz a la etapa de implementación de dominio, quien luego, debe construir las interfaces diseñadas. [24].

Ante la inserción de un componente de la *estructura* se deberá analizar:

- Componentes reutilizables: debido a que la estructura determina la descomposición de los componentes que son posibles y válidos para todas las aplicaciones de la línea de productos [23, 31] se deberá construir el nuevo componente reutilizable en la etapa de Implementación del domino.

5.3.5. Entre Diseño del Dominio e Implementación del Dominio para el escenario de evolución de modificación

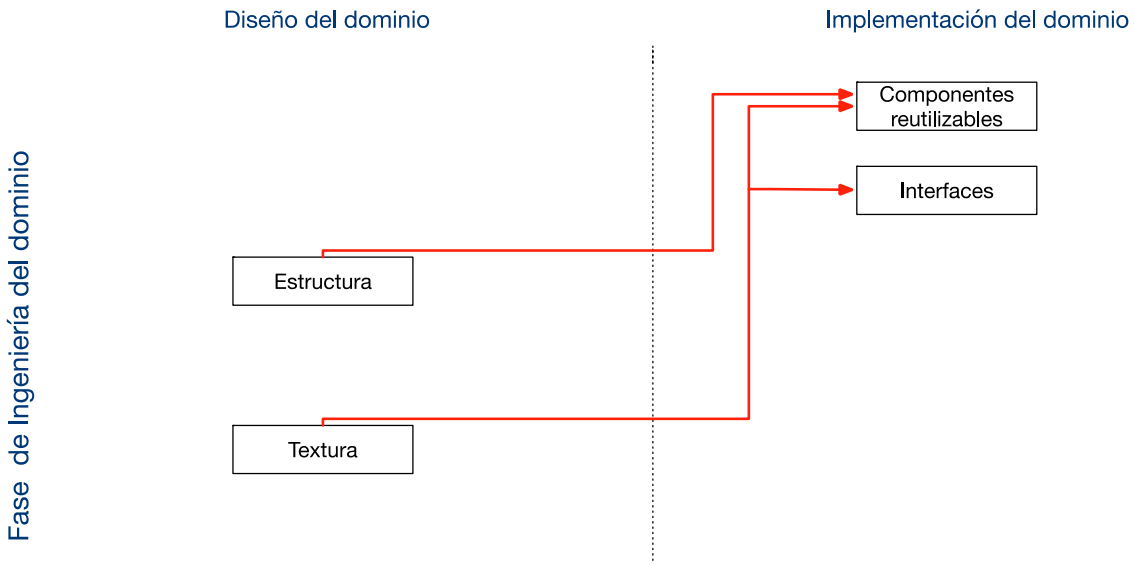

Figura 48: Trazabilidad horizontal entre Diseño del dominio e Implementación del dominio, para el escenario de evolución de modificación

En la figura 48 se observa la trazabilidad horizontal entre las etapas de Diseño del dominio e Implementación del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de una *textura* se debe analizar:

- Componentes reutilizables: debido a que pueden ser modificadas reglas de codificación y mecanismos generales para el proceso de codificación. Por lo tanto, estos cambios deben ser considerados en la implementación de los componentes reutilizables para asegurar la reusabilidad de todo el diseño y los artefactos de implementación [24].

- Interfaces: debido a que la textura provee, entre otras cosas, el diseño de interfaz a la etapa de implementación de dominio, se deberán modificar las interfaces considerando los cambios en el diseño de la interfaz. [24].

Ante la modificación de la *estructura* se deberá analizar:

- Componentes reutilizables: debido a que la estructura determina la descomposición de los componentes que son posibles y válidos para todas las aplicaciones de la línea de productos [22, 23], se deberá modificar el o los componentes reutilizables considerando los cambios que se realizaron en la etapa de Diseño del dominio.

5.3.6. Entre Diseño del Dominio e Implementación del Dominio para el escenario de evolución de eliminación

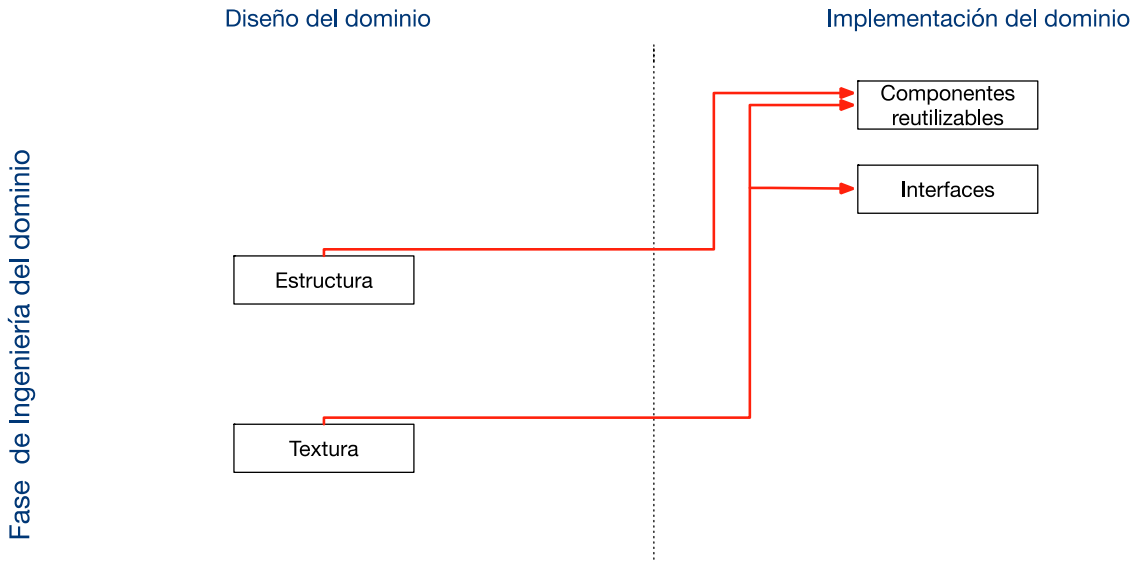

Figura 49: Trazabilidad horizontal entre Diseño del dominio e Implementación del dominio, para el escenario de evolución de eliminación

En la figura 49 se observa la trazabilidad horizontal entre las etapas de Diseño del dominio e Implementación del dominio, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de una *textura* se debe analizar:

- Componente reutilizable: debido a que pueden ser eliminadas algunas reglas de codificación y mecanismos generales para el proceso de codificación, se deberá actualizar el componente reutilizable, considerando los cambios realizados en la textura [24].
- Interfaces: la textura provee, entre otras cosas, el diseño de interfaz a la etapa de implementación de dominio. Por lo tanto, si se realiza una eliminación en el diseño de la interfaz proporcionada por la textura, se deberá actualizar las interfaces en la etapa de Implementación del dominio [24].

Ante la eliminación de un componente de la *estructura* se deberá analizar:

- Componentes reutilizables: debido a que la estructura determina la descomposición de los componentes que son posibles y válidos para todas las aplicaciones de la línea de productos [23, 27], se deberá eliminar el componente relacionado con el componente eliminado de la estructura.

5.3.7. Trazabilidad horizontal entre Pruebas del Dominio con cada una de las etapas de la fase de Ingeniería del Dominio

El objetivo principal de las pruebas del dominio es validar los artefactos creados en la etapa de Implementación [24]. Sin embargo, se deben validar todos los artefactos relevantes de cada una de las etapas previas. Es por esto que la trazabilidad horizontal será entre la etapa de Pruebas del dominio y cada una de las etapas anteriores a ésta, es decir, Ingeniería de requisitos del dominio, Diseño del dominio e Implementación del dominio.

Con una retroalimentación en cada etapa, será posible entregar artefactos de calidad a las etapas que dependen de la anterior [20].

La trazabilidad horizontal se deberá realizar de igual forma para cualquiera de los tres escenarios definidos para este proyecto (inserción, modificación y eliminación).

La figura 50 representa la trazabilidad horizontal entre la etapa de Pruebas del dominio y cada una de las etapas correspondientes a la fase de Ingeniería del dominio, es decir, Ingeniería de requisitos del dominio, Diseño del dominio e Implementación del dominio.

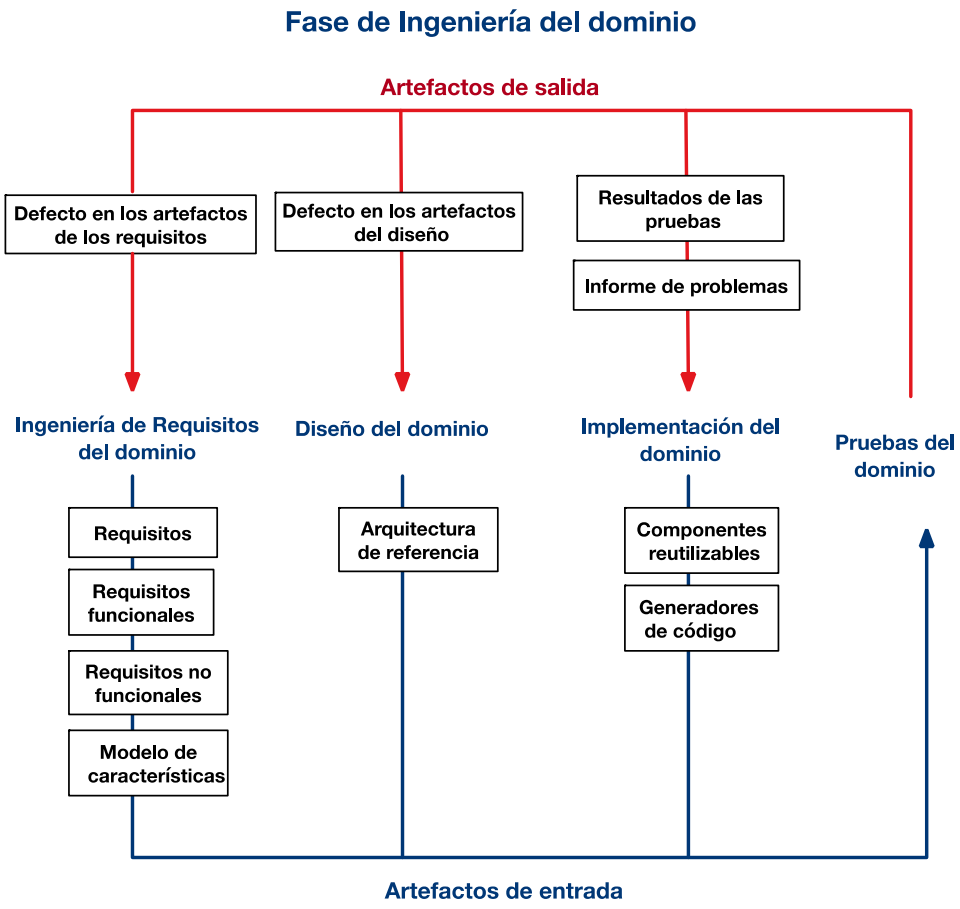

Figura 50: Modelo de trazabilidad horizontal entre la etapa de Pruebas del dominio y cada una de las etapas de la Fase de Ingeniería del dominio

La trazabilidad horizontal entre la etapa de Pruebas del dominio con cada una de las etapas de la Fase de Ingeniería del dominio es explicada a continuación:

5.3.7.1. Trazabilidad horizontal entre las etapas de Pruebas del dominio e Ingeniería de requisitos del dominio

La entrada derivada de la Ingeniería de requisitos del dominio para la etapa de Pruebas del dominio consiste en los artefactos de requisitos del dominio y el modelo de características. Los artefactos de requisitos de dominio contienen a los requisitos comunes y variables, considerando los requisitos funcionales y los no funcionales del sistema. Las etapa de Pruebas del dominio utiliza los artefactos de la Ingeniería de requisitos del dominio para desarrollar las pruebas del sistema [20].

Los artefactos encontrados en los requisitos se informan a la ingeniería de requisitos del dominio, de modo que puedan ser corregidos. Por consiguiente, las pruebas del dominio contribuyen a la validación de los requisitos del dominio, asegurando la calidad general de la línea de productos de software [20].

5.3.7.2. Trazabilidad horizontal entre Pruebas del dominio y Diseño del dominio

La entrada derivada de la etapa de Diseño del dominio para las Pruebas del dominio consiste en la Arquitectura de referencia. Se utiliza la Arquitectura de referencia como referencia de prueba para validar las interacciones entre los componentes de la arquitectura. Tales interacciones son, por ejemplo, entre los componentes que se encuentran en la estructura arquitectónica [20]. Cabe destacar que las pruebas realizadas en la etapa de diseño no validan todas las interacciones entre los componentes, ya que pueden existir componentes que no se desarrollan en la fase de Ingeniería del dominio, si no, que se desarrollan en la fase de Ingeniería de la aplicación.

Los defectos encontrados en los artefactos de diseño del dominio, tales como incompletitud y ambigüedad son reportados a la etapa de diseño del dominio para su corrección.

5.3.7.3. Trazabilidad horizontal entre las etapas Pruebas del dominio e Implementación del dominio

La entrada derivada de la etapa de Implementación del dominio para las Pruebas del dominio consiste en los componentes e interfaces reutilizables y los generadores de código. Cabe destacar que solo se validan los componentes e interfaces reutilizables creadas en la fase de Ingeniería del dominio [20] y no las creadas en la fase de Ingeniería de la aplicación.

Las pruebas del dominio proporcionan a la etapa de Implementación del dominio, los resultados de las pruebas, incluyendo la aceptación o el rechazo, además de los informes de problemas correspondientes. Los informes de problemas capturan las desviaciones observadas del comportamiento esperado de los artefactos de implementación. Los defectos encontrados en las descripciones de la interfaz obstaculizan el diseño del caso de prueba, por lo que debe ser corregido antes de que la prueba pueda ser completada [20].

5.4. Trazabilidad horizontal en fase de Ingeniería de la aplicación

5.4.1. Trazabilidad horizontal entre las etapas de Requisitos de la aplicación y Diseño de la aplicación, para el escenario de evolución de inserción

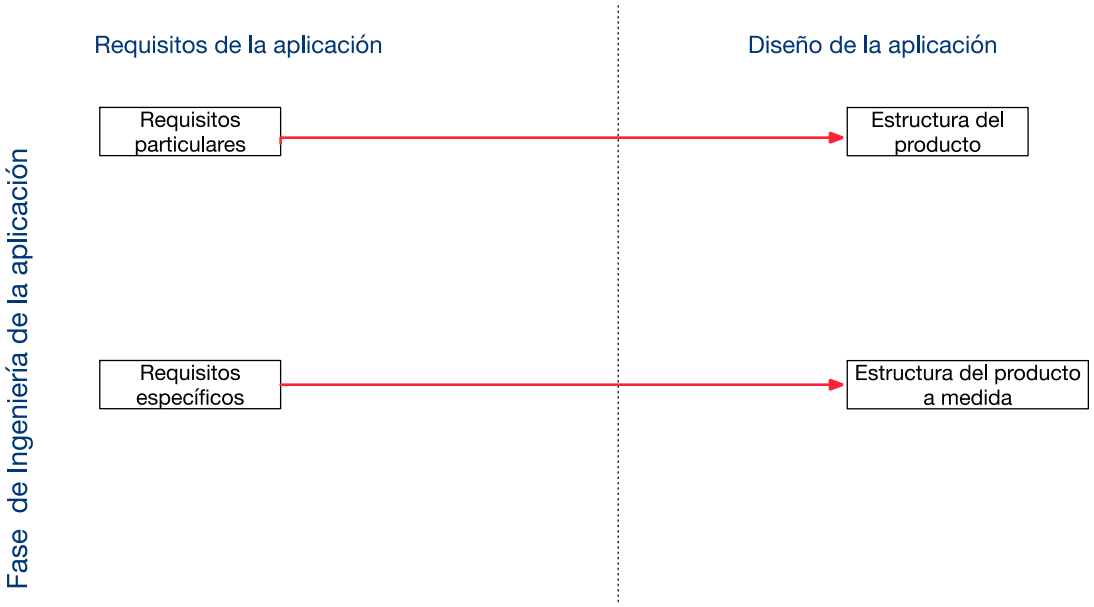

Figura 51: Trazabilidad horizontal entre la etapa de Requisitos de la aplicación y la etapa de Diseño de la aplicación, para el escenario de evolución de inserción

En la figura 51 se observa la trazabilidad horizontal entre las etapas de Requisitos de la aplicación y Diseño de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de un *requisito particular* se deberá analizar:

- Estructura del producto: se deberá agregar el componente de la estructura particular que nace a partir del nuevo requisito particular, el cual es instanciado desde la etapa de Ingeniería de requisitos del dominio [20].

Ante la inserción de un *requisito específico* se deberá analizar:

- Estructura del producto a medida: se deberá agregar el componente de la estructura correspondiente al requisito específico creado para un producto en particular [20].

5.4.2. Trazabilidad horizontal entre las etapas de entre Requisitos de la aplicación y Diseño de la aplicación, para el escenario de evolución de modificación

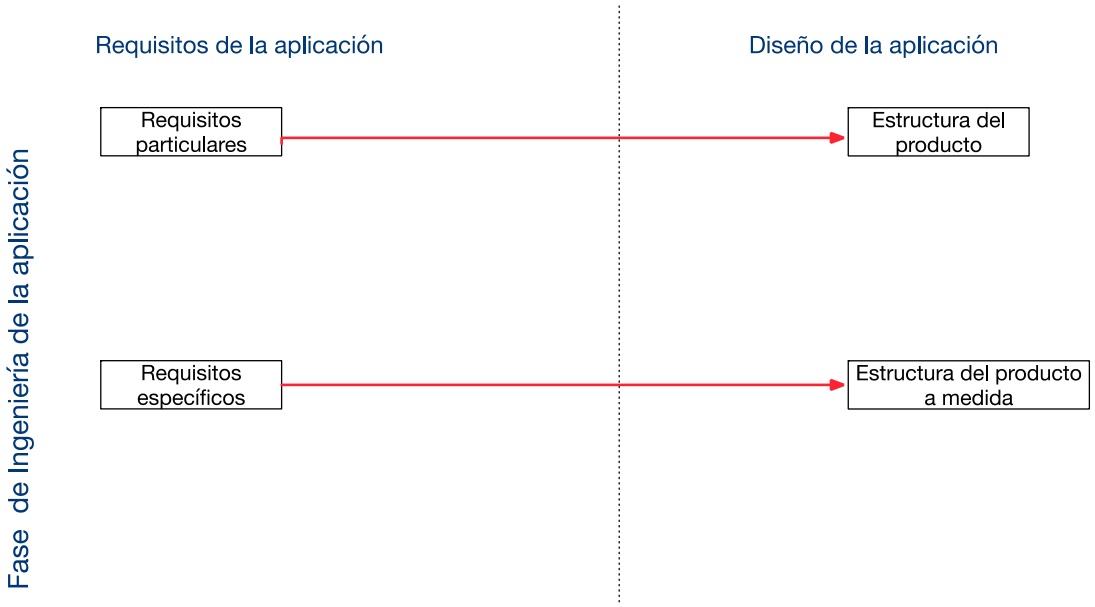

Figura 52: Trazabilidad horizontal entre la etapas de Requisitos de la aplicación y de Diseño de la aplicación, para el escenario de evolución de modificación

En la figura 52 se observa la trazabilidad horizontal entre las etapas de Requisitos de la aplicación y Diseño de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de un *requisito particular* se deberá analizar:

- Estructura del producto: se deberá modificar el componente de la estructura del producto asociada al requisito particular modificado [20].

Ante la modificación de un *requisito específico* se deberá analizar:

- Estructura del producto a medida: se deberá modificar el componente de la estructura del producto a medida, correspondiente al requisito específico modificado [20].

5.4.3. Trazabilidad horizontal entre las etapas de Requisitos de la aplicación y la de Diseño de la aplicación, para el escenario de evolución de eliminación

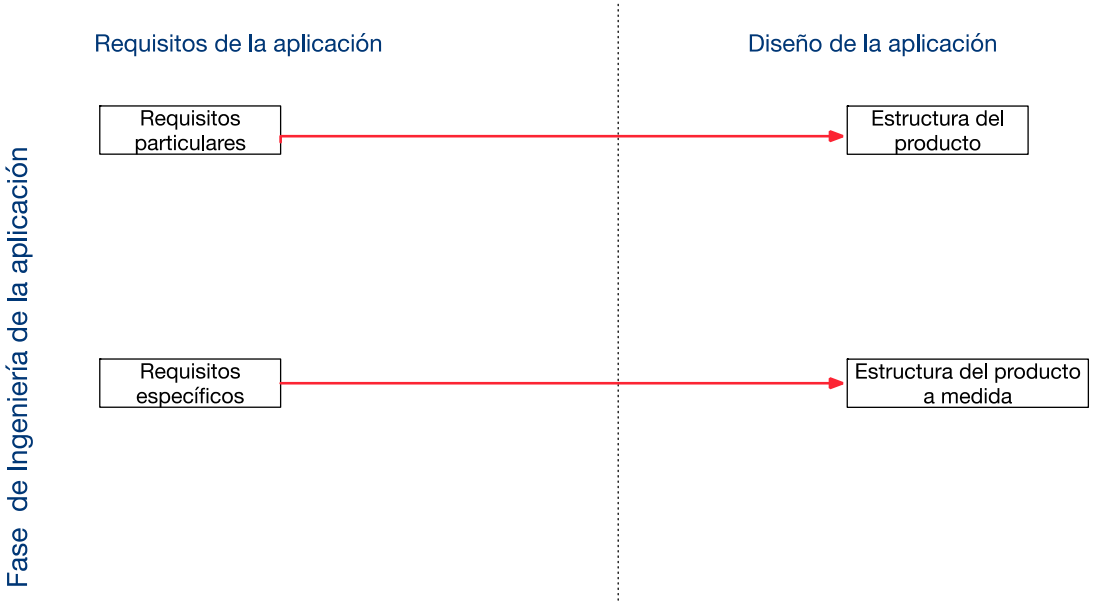

Figura 53: Trazabilidad horizontal entre la etapa de Requisitos de la aplicación y la etapa de Diseño de la aplicación, para el escenario de evolución de eliminación

En la figura 53 se observa la trazabilidad horizontal entre las etapas de Requisitos de la aplicación y Diseño de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de un *requisito particular* se deberá analizar:

- Estructura del producto: se deberá eliminar el componente de la estructura del producto asociada al requisito particular eliminado [20].

Ante la eliminación de un *requisito específico* se deberá analizar:

- Estructura del producto a medida: se deberá eliminar el componente de la estructura del producto a medida, correspondiente al requisito específico eliminado [20].

5.4.4. Entre Diseño de la aplicación e Implementación de la aplicación, escenario de evolución de inserción

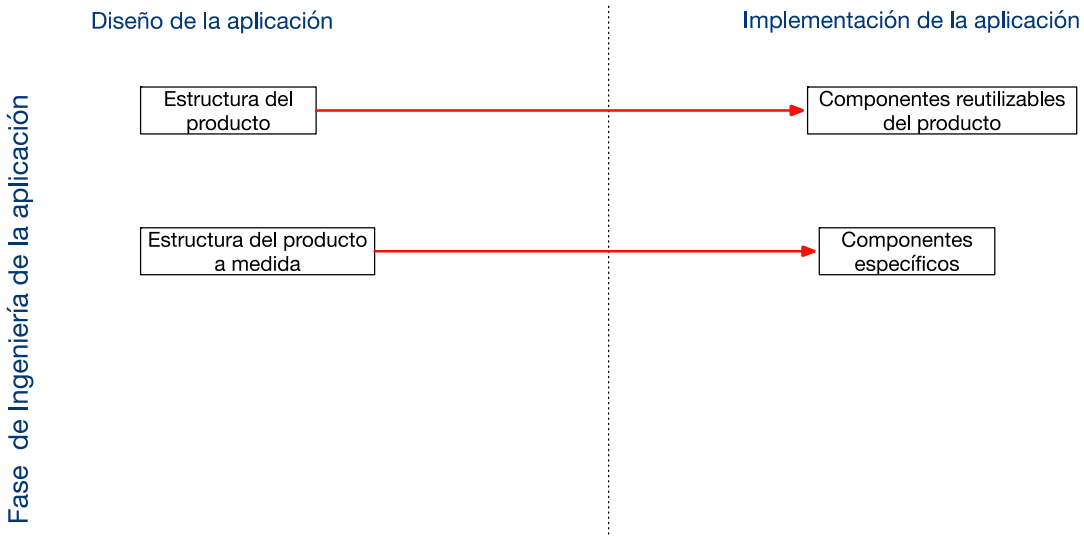

Figura 54: Trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, para el escenario de evolución de inserción.

En la figura 54 se observa la trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de algún componente en la *estructura del producto* se deberá analizar:

- Componente reutilizable del producto: debido a que la estructura del producto determina la estructura de la aplicación que se va a construir, se deberá instanciar el nuevo componente reutilizable que dependía del componente de la estructura agregado [20].

Ante la inserción de un componente de la *estructura del producto a medida* se deberá analizar:

- Componente específico: como la estructura del producto a medida es la entrada para el desarrollo de los componentes específicos, se deberá construir el componente específico, considerando el nuevo componente de la Estructura del producto a medida [20].

5.4.5. Trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, para el escenario de evolución de modificación

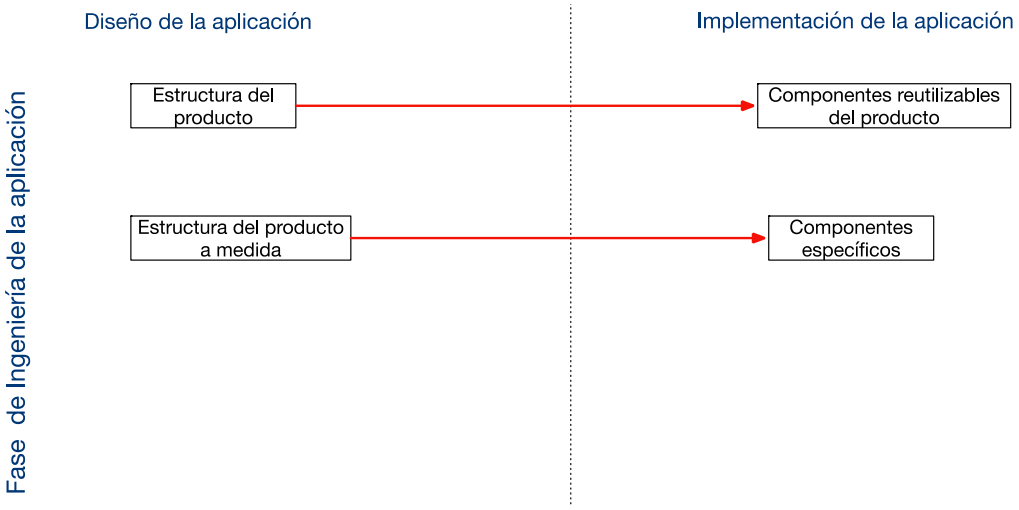

Figura 55: Trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, para el escenario de evolución de modificación

En la figura 55 se observa la trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de algún componente en la *estructura del producto* se deberá analizar:

- Componente reutilizable del producto: debido a que la estructura del producto determina la estructura de la aplicación que se va a construir, se deberá modificar el componente reutilizable que dependía del componente de la estructura modificado [20].

Ante la modificación de un componente de la *estructura del producto a medida* se deberá analizar:

- Componente específico: como la estructura del producto a medida es la entrada para el desarrollo de los componentes específicos, se deberá modificar el componente específico del producto, considerando el componente de la Estructura del producto a medida modificado [20].

5.4.6. Trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, para el escenario de evolución de eliminación

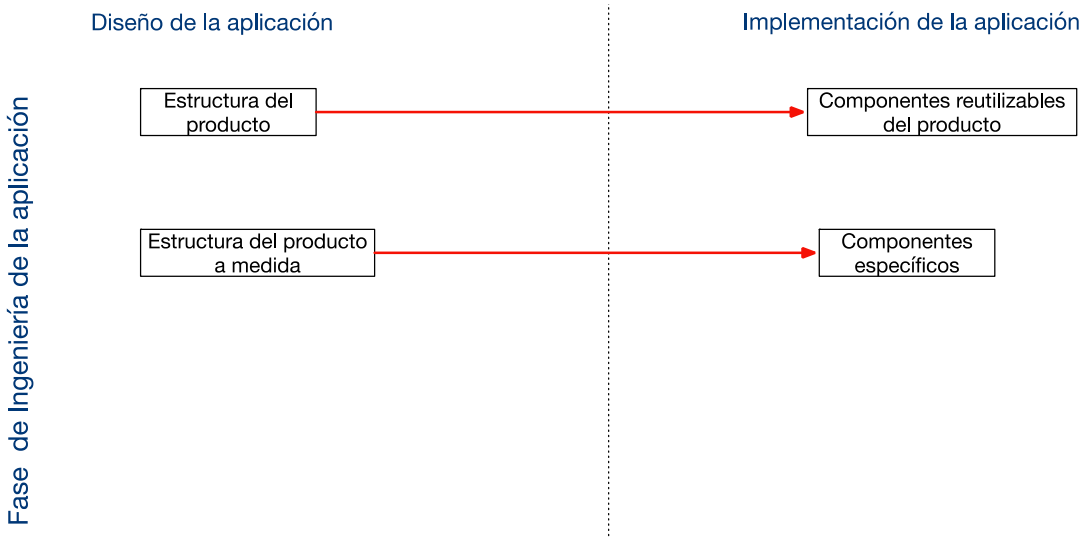

Figura 56: Trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, para el escenario de evolución de eliminación

En la figura 56 se observa la trazabilidad horizontal entre las etapas de Diseño de la aplicación e Implementación de la aplicación, representada por las flechas de color rojo. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de algún componente en la *estructura del producto* se deberá analizar:

- Componente reutilizable: se deberá eliminar el componente reutilizable que dependía del componente de la estructura eliminado [20].

Ante la eliminación de un componente de la *estructura del producto a medida* se deberá analizar:

- Componente específico: se deberá eliminar el componente específico, considerando el componente de la estructura del producto a medida eliminado [20].

5.4.7. Trazabilidad horizontal entre la etapa de Pruebas de la aplicación y cada una de las etapas de la Fase de la Ingeniería de la aplicación

Las pruebas de la aplicación, las cuales reutilizan los artefactos creados en la etapa de pruebas del dominio, son un complemento a las actividades de las pruebas del dominio [30].

La trazabilidad horizontal será entre la etapa de Pruebas de la aplicación y cada una de las etapas previas en la fase de Ingeniería de la aplicación, es decir, Ingeniería de la aplicación, Diseño de la aplicación e Implementación de la aplicación.

La trazabilidad horizontal se deberá realizar de igual forma para cualquiera de los tres escenarios definidos para este proyecto (inserción, modificación y eliminación). La figura 57 representa la trazabilidad horizontal entre la etapa de Pruebas de la aplicación y cada una de las etapas correspondientes a la fase de Ingeniería de la aplicación, es decir, Requisitos de la aplicación, Diseño de la aplicación e Implementación de la aplicación.

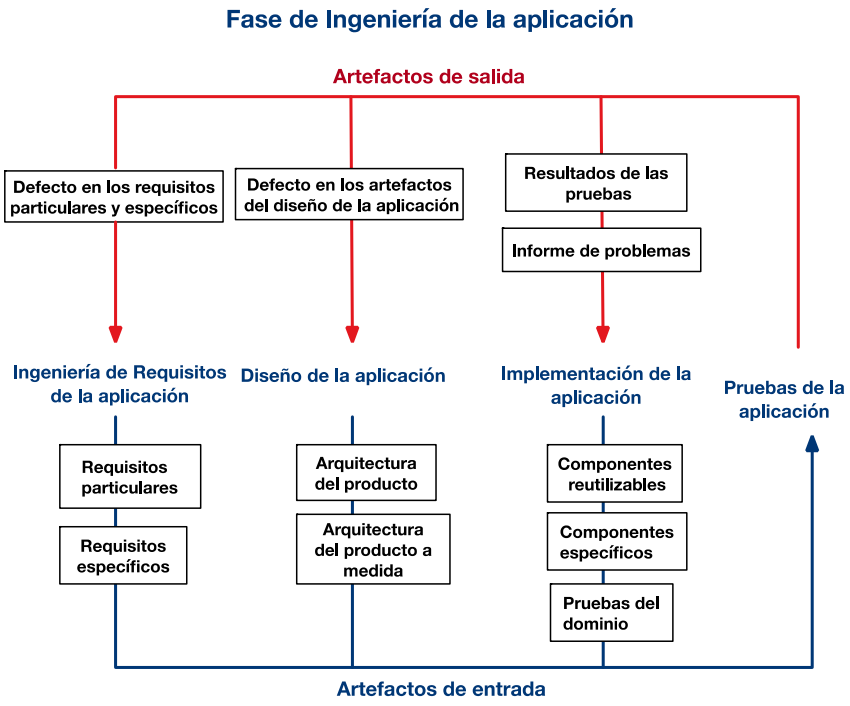

Figura 57: Modelo de trazabilidad horizontal para la etapa de Pruebas de la aplicación con cada etapa de la fase de Ingeniería de la aplicación.

La trazabilidad horizontal entre la etapa de Pruebas de la aplicación con cada una de las etapas de la Fase de Ingeniería de la aplicación es explicada a continuación.

5.4.7.1. Trazabilidad horizontal entre las etapas de Pruebas de la aplicación e Ingeniería de la aplicación

Las pruebas de la aplicación validan el producto creado en base al documento de requisitos de la aplicación [30]. Estas pruebas de la aplicación emplean como referencia de prueba a los requisitos de la aplicación (requisitos particulares y requisitos específicos). La validación debe garantizar que no se ha omitido ningún requisito particular o específico.

La creación de casos de prueba es al mismo tiempo una validación de los requisitos de aplicación. Los requisitos defectuosos, tales como requisitos ambiguos o incompletos deben ser informados a la etapa de Requisitos de la aplicación.

5.4.7.2. Trazabilidad horizontal entre las etapas de Pruebas de la aplicación y Diseño de la aplicación

La entrada de la etapa de Diseño de la aplicación para las pruebas del dominio consiste en la arquitectura del producto y la arquitectura del producto a medida. Los artefactos de prueba reutilizables están disponibles solo para los componentes de la arquitectura del producto, ya que éstos fueron instanciados desde la etapa de diseño del dominio [30].

Siempre que un tester de la aplicación no pueda determinar los datos requeridos para un caso de prueba, se ha detectado un incompletitud o ambigüedad. Cualquier defecto encontrado en los artefactos de diseño mencionados anteriormente, debe ser reportado a la etapa de Diseño de la aplicación.

5.4.7.3. Trazabilidad horizontal entre las etapas de Pruebas de la aplicación e Implementación de la aplicación

Las Pruebas de la aplicación validan tanto los componentes reutilizables, como los componentes específicos, pero también repite las pruebas ya realizadas en las pruebas del dominio. Las pruebas de la aplicación reportan todos los resultados de las pruebas a los componentes reutilizables y específicos, junto con un informe de problemas. Los resultados de las pruebas capturan los casos de prueba que se han realizado y si el objeto bajo prueba falló o aprobó la prueba [30]. Cualquier defecto encontrado en los artefactos de prueba de la etapa de Implementación de la aplicación deben ser reportados a esta etapa.

5.5. Trazabilidad vertical

5.5.1. Entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, para el escenario de evolución de inserción

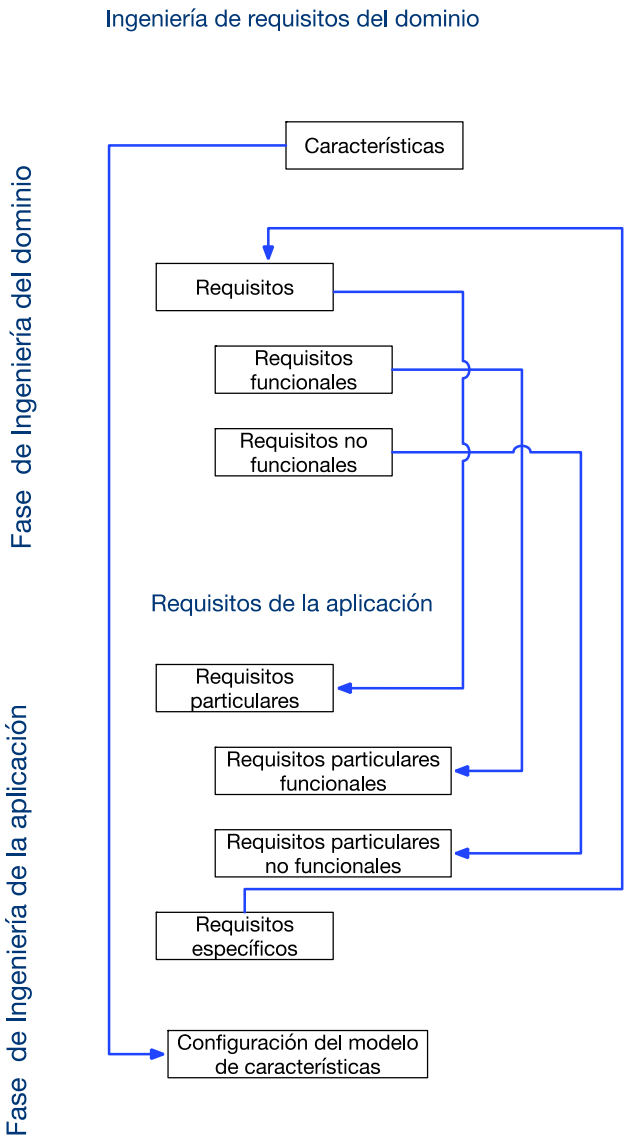

Figura 58: Trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, para el escenario de evolución de inserción

En la figura 58 se observa la trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de una *característica* se debe analizar:

- Configuración del modelo de característica: debido a que este contempla las características definidas en la Ingeniería del dominio, eligiendo algunas de estas para instanciarlas a un producto en particular [22]. Por lo tanto, si se agrega una nueva característica al modelo de característica, se deberá instanciar esa nueva característica a través de la configuración del modelo de característica.

Ante la inserción de un *requisito* se deberá analizar:

- Requisito particular: se deberá instanciar a la aplicación correspondiente, el nuevo requisito creado en la etapa de Ingeniería de requisitos del dominio [24].

Ante la inserción de un *requisito funcional* se deberá analizar:

- Requisito particular funcional: se deberá instanciar a la aplicación correspondiente, el nuevo requisito funcional creado por la etapa de ingeniería de requisitos del dominio [24].

Ante la inserción de un *requisito no funcional* se deberá analizar:

- Requisito particular no funcional: se deberá instanciar a la aplicación correspondiente, el nuevo requisito no funcional creado por la etapa de ingeniería de requisitos del dominio [24].

Ante la inserción de un *requisito específico* se deberá analizar:

- Requisito: debido a que existe retroalimentación entre las fases de Ingeniería del dominio e Ingeniería de la aplicación, puede ocurrir que un requisito específico se transforme en un requisito reutilizable, es decir, que pueda ser usado por más

de un producto en la LPS. Por lo tanto, se debe analizar si el requisito específico pasa a ser parte de la etapa de Ingeniería de requisitos del Dominio.

5.5.2. Trazabilidad vertical entre Ingeniería de requisitos del Dominio y Requisitos de la Aplicación, para el escenario de evolución de modificación

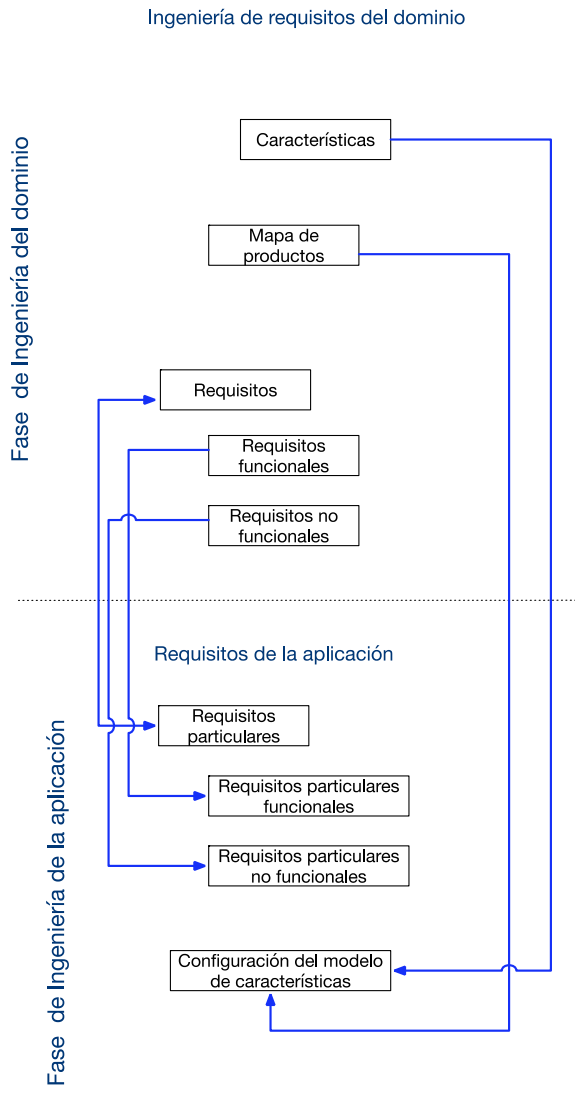

Figura 59: Trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, para el escenario de evolución de modificación

En la figura 59 se observa la trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de un *requisito* se deberá analizar:

- Requisito particular: al modificar un requisito de la Ingeniería del dominio, se deberá modificar el requisito particular, ya que es el mismo requisito pero instanciado a un producto particular en la etapa de Ingeniería de la aplicación [22].

Ante la modificación de un *requisito particular* se deberá analizar:

- Requisito: debido a la retroalimentación que existe entre la Ingeniería de Requisitos del dominio y la Ingeniería de la aplicación [24], puede existir el caso en que un requisito particular no satisfaga completamente al producto, por lo que se deberá notificar a la Ingeniería de requisitos del dominio para hacer la respectiva modificación del requisito [32].

Ante la modificación de un *requisito funcional* se deberá analizar:

- Requisito particular funcional: se deberá modificar el requisito particular funcional, debido a que es el mismo requisito de la Ingeniería de requisitos del dominio, instanciado a un producto particular [22].

Ante la modificación de un *requisito no funcional* se deberá analizar:

- Requisito particular no funcional: se deberá modificar el requisito particular no funcional, debido a que es el mismo requisito de la Ingeniería de requisitos del dominio, instanciado a un producto particular [22].

Ante la modificación del *mapa de productos* se deberá analizar:

- Configuración del modelo de características: debido a que el mapa de productos es el que define las características que tendrá cada producto en particular, se deberá

actualizar la configuración del modelo de características para el producto involucrado en la modificación del mapa de productos [20].

Ante la modificación de una *característica* se deberá analizar:

- Configuración del modelo de características: se deberá modificar la configuración del modelo de características, debido a que éste comprende las características del modelo de características elegidas para un producto en particular [22].

5.5.3. Trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, para el escenario de evolución de eliminación

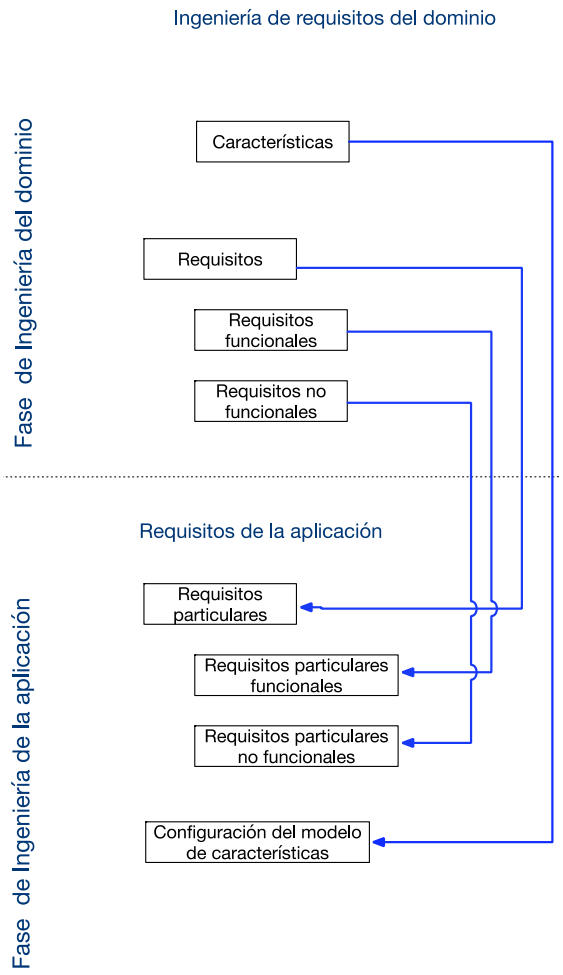

Figura 60: Trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, para el escenario de evolución de eliminación

En la figura 60 se observa la trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de una *característica* se deberá analizar:

- Configuración del modelo de características: se deberá actualizar la configuración del modelo de características, eliminado aquella que fue eliminada del modelo de características en la etapa de Ingeniería de requisitos del dominio [22].

Ante la eliminación de un *requisito* se deberá analizar:

- Requisito particular: al eliminar un requisito de la Ingeniería del dominio, se deberá modificar el requisito particular, ya que es el mismo requisito pero instanciado a un producto particular en la etapa de Ingeniería de la aplicación [22].

Ante la eliminación de un *requisito funcional* se deberá analizar:

- Requisito particular funcional: se deberá eliminar el requisito particular funcional, debido a que es el mismo requisito de la Ingeniería de requisitos del dominio, instanciado a un producto particular [22].

Ante la eliminación de un *requisito no funcional* se deberá analizar:

- Requisito particular no funcional: se deberá eliminar el requisito particular no funcional, debido a que es el mismo requisito de la Ingeniería de requisitos del dominio, instanciado a un producto particular [22].

5.5.4. Trazabilidad vertical entre las etapas de Diseño del domino y Diseño de la aplicación, para el escenario de evolución de inserción

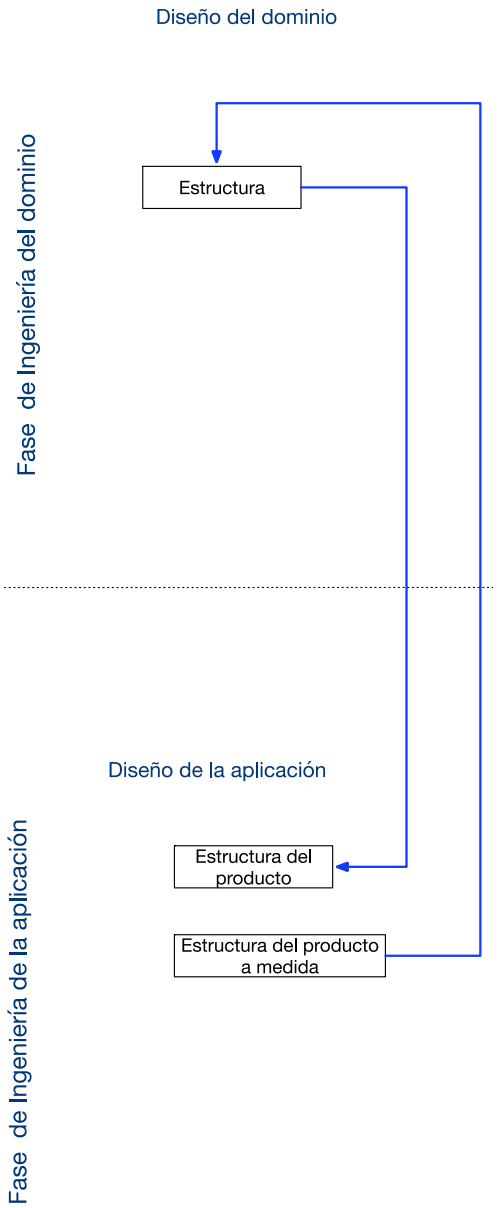

Figura 61: Trazabilidad vertical entre las etapas de Diseño del dominio y Diseño de la aplicación, para el escenario de evolución de inserción

En la figura 61 se observa la trazabilidad vertical entre las etapas de Ingeniería de requisitos del dominio y Requisitos de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de un componente en la *estructura* se deberá analizar:

- Estructura del producto: como la estructura del producto, consiste en la instanciación de ciertos componentes de la estructura, se deberá agregar el nuevo componente para ser instanciado a la estructura del producto particular [29].

Ante la inserción de la *estructura del producto a medida* se deberá analizar:

- Estructura: Se deberá analizar si la estructura del producto a medida será parte de un solo producto particular o si formará parte de la arquitectura de referencia. Esto último significa iniciar una evolución en la línea de productos, por lo tanto, requiere de un esfuerzo adicional para la reingeniería [29].

5.5.5. Trazabilidad vertical entre las etapas de Diseño del domino y Diseño de la aplicación, para el escenario de evolución de modificación

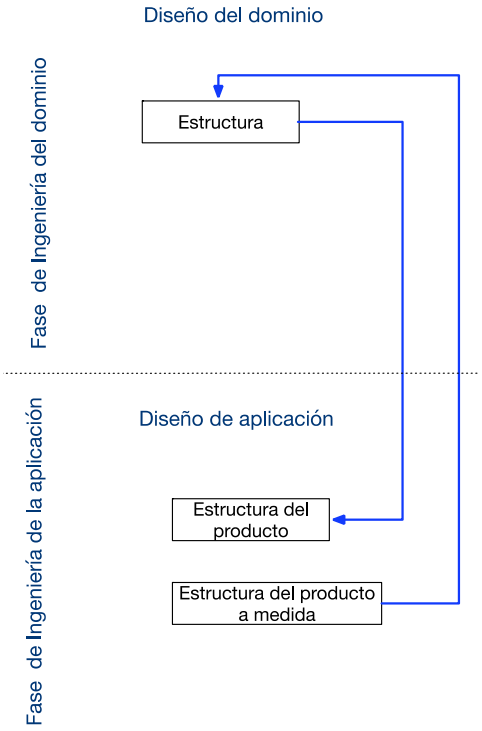

Figura 62: Trazabilidad vertical entre las etapas de Diseño del Dominio y Diseño de Aplicación, para el escenario de evolución de modificación

En la figura 62 se observa la trazabilidad vertical entre las etapas de Diseño del Dominio y Diseño de la Aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de la *estructura* de deberá analizar:

- Estructura del producto: se deberá modificar la estructura del producto, debido a que ésta consiste en la instanciación de ciertos componentes de la Estructura del dominio [29].

Ante la modificación de la *estructura del producto* se deberá analizar:

- Estructura: Debido a que existe retroalimentación entre las fases de Ingeniería del dominio e Ingeniería de la aplicación, se deberá notificar a la etapa de Diseño del dominio, si es que la estructura del dominio no satisface completamente un producto en particular [29].

5.5.6. Trazabilidad vertical entre las etapas de Diseño del Domino y Diseño de la Aplicación, para el escenario de evolución de eliminación

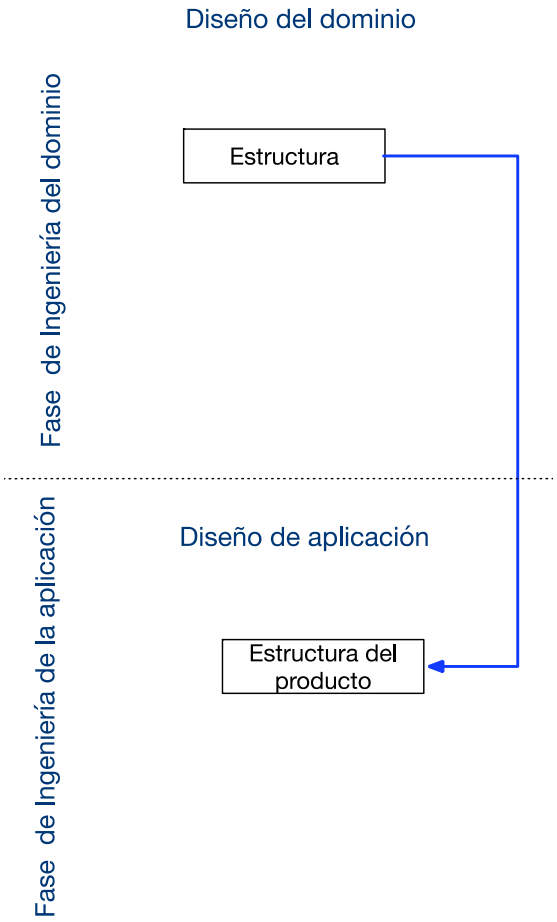

Figura 63: Trazabilidad vertical entre las etapas de Diseño del Dominio y Diseño de la Aplicación, para el escenario de evolución de eliminación

En la figura 63 se observa la trazabilidad vertical entre las etapas de Diseño del dominio y Diseño de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de algún componente de la *estructura* se deberá analizar:

- Estructura del producto: se deberá modificar la estructura del producto, considerando aquellos componentes de la estructura del dominio eliminados, ya que estos componentes son los que serán instanciados a un producto en particular.

5.5.7. Trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de inserción

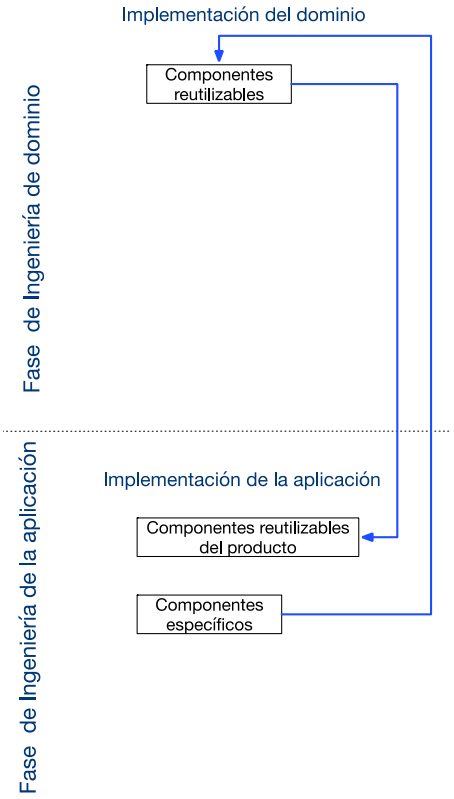

Figura 64: Trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de inserción

En la figura 64 se observa la trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de un *componente reutilizable* se deberá analizar:

- Componente reutilizable del producto: debido a que la Implementación de la aplicación instancia ciertos componentes reutilizables implementados en la etapa de Implementación del dominio, para la realización de un producto en particular, se deberá agregar dichos componentes reutilizables hacia un producto en particular [20].

Ante la inserción de algún *componente específico* se deberá analizar:

- Componente reutilizable: se deberá analizar si el nuevo componente específico formará parte de los componentes reutilizables dentro de la fase de Ingeniería del dominio, con el fin de ser reutilizado por más de un producto de la LPS [20].

5.5.8. Trazabilidad vertical entre la etapa de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de modificación

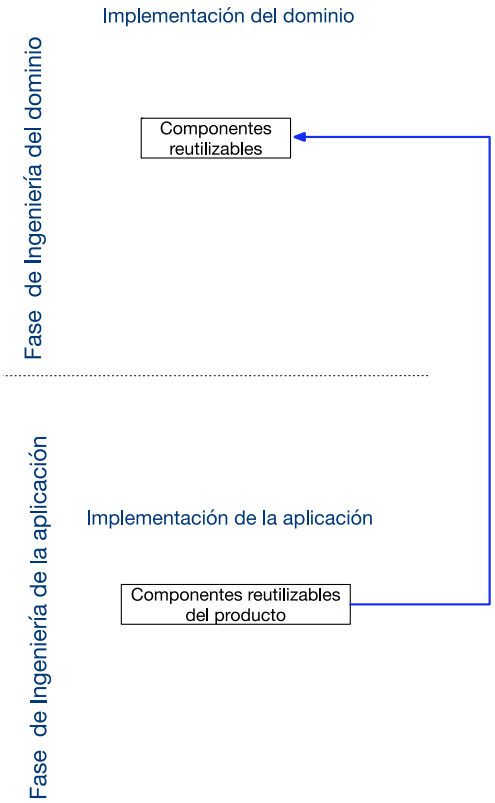

Figura 65: Trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de modificación.

En la figura 65 se observa la trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación de un *componente reutilizable del producto se* deberá analizar:

- Componente reutilizable: Puede ser detectado en algún componente reutilizable del producto algún defecto o simplemente que el componente reutilizable no satisfaga

completamente lo que se desea para el producto. Gracias a esta retroalimentación entre ambas fases, se podrá mejorar un componente reutilizable para lograr satisfacer la gama completa de productos [20].

5.5.9. Trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de eliminación

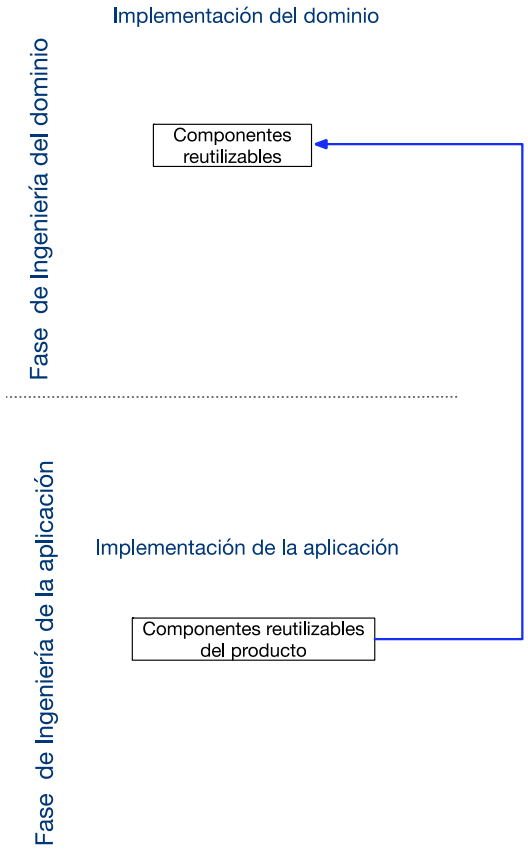

Figura 66: Trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, para el escenario de evolución de eliminación

En la figura 66 se observa la trazabilidad vertical entre las etapas de Implementación del dominio e Implementación de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de algún *componente reutilizable* del producto se deberá analizar:

- Componente reutilizable: se deberá eliminar el componente reutilizable en el producto que consideraba el componente instanciado desde la etapa de Implementación del dominio hacia la etapa de Implementación de la aplicación [20].

5.5.10. Trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de inserción

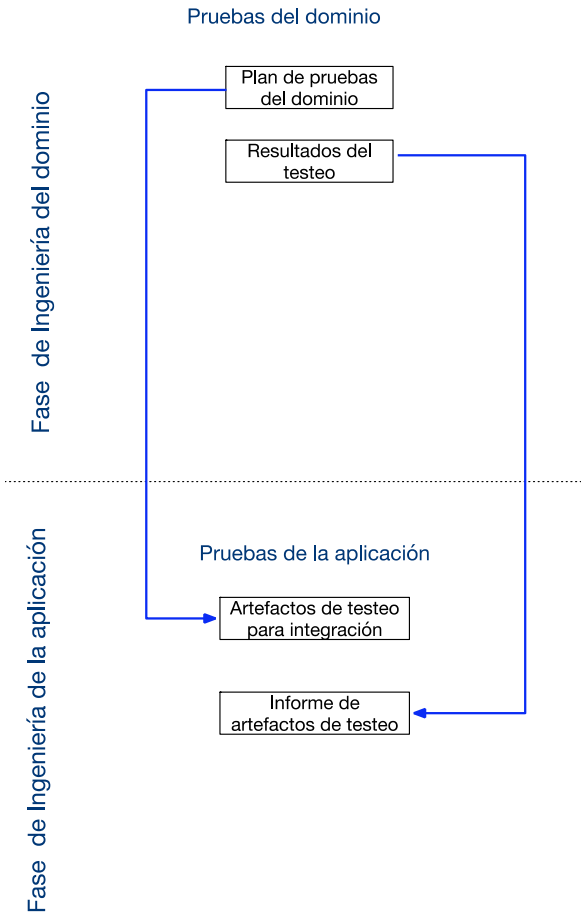

Figura 67: Trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de inserción

En la figura 67 se observa la trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la inserción de algún artefacto dentro de las etapas mencionadas.

Ante la inserción de algún artefacto de testeo dentro del *plan de pruebas del dominio* se deberá analizar:

- Artefactos de testeo para integración: debido a que en el plan de pruebas del dominio se definen todos los artefactos de pruebas que se utilizarán en la etapa de Pruebas de la aplicación [24].

Ante la inserción de algún *resultado del testeo* se deberá analizar:

- Informe de artefactos de testeo: debido a que el Informe de artefactos de testeo comprende todos los defectos en los artefactos de testeo que se encuentran en el artefacto Resultado de testeo [24].

5.5.11. Entre Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de modificación

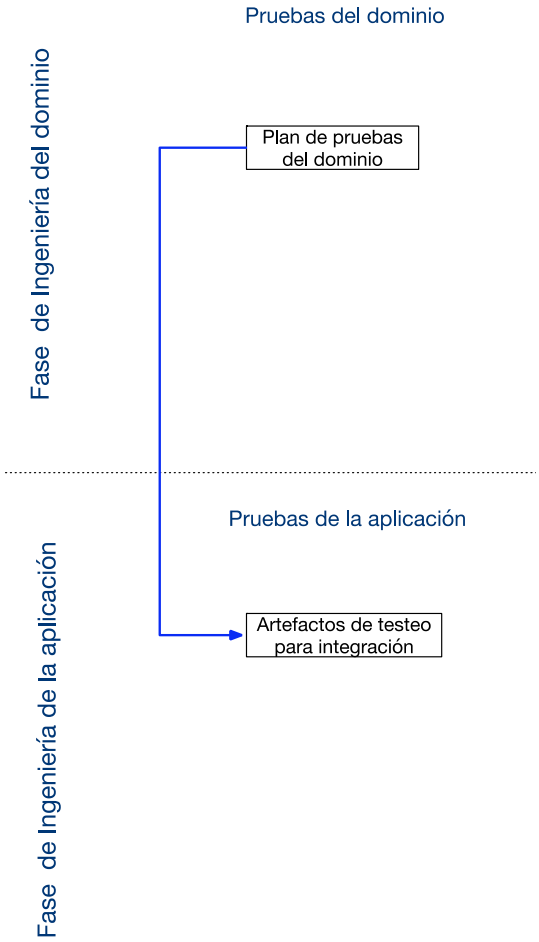

Figura 68: Trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de modificación

En la figura 68 se observa la trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la modificación de algún artefacto dentro de las etapas mencionadas.

Ante la modificación en el *plan de pruebas del dominio* se deberá analizar:

- Artefactos de testeo para integración: debido a que los artefactos de testeo para integración son los artefactos definidos en el Plan de pruebas del dominio, los

cuales se utilizan en las Pruebas de la aplicación. Por lo tanto, se deberá actualizar el artefacto de testeo para integración [24].

5.5.12. Trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de eliminación

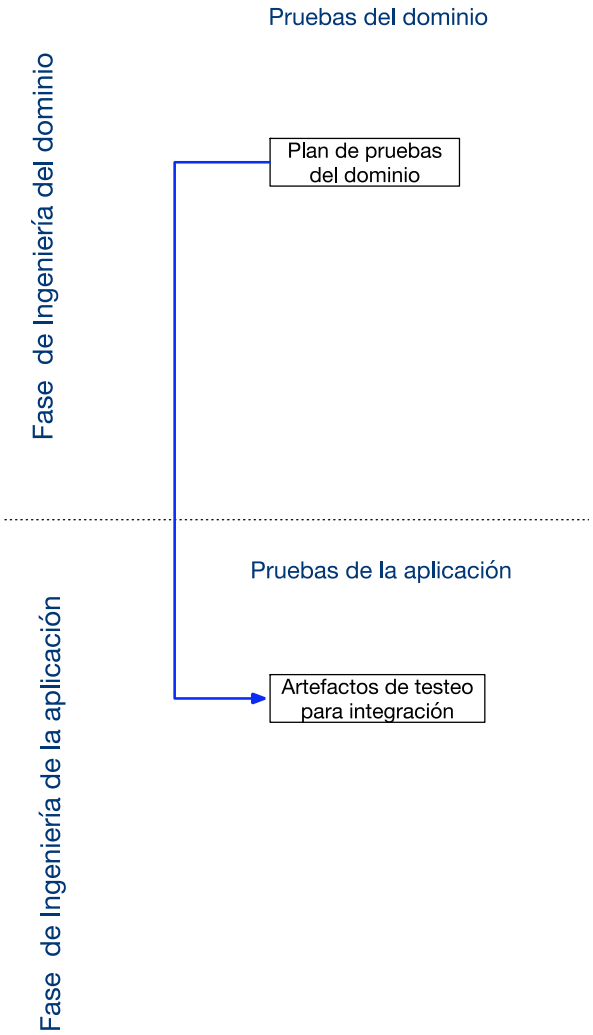

Figura 69: Trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de eliminación

En la figura 69 se observa la trazabilidad vertical entre las etapas de Pruebas del dominio y Pruebas de la aplicación, representada por las flechas de color azul. A continuación, se explican los eventos correspondientes a la eliminación de algún artefacto dentro de las etapas mencionadas.

Ante la eliminación de un *plan de pruebas del dominio* se deberá analizar:

- Artefactos de testeo para integración: si se elimina un artefacto de prueba definido por el plan de pruebas de dominio, se deberá eliminar en los artefactos de testeo para integración, debido a que éstos instancian desde la etapa de Pruebas del dominio [21].

5.6. Modelo de trazabilidad, vista completa

El modelo de trazabilidad se presenta de acuerdo a los escenarios de evolución definidos previamente (inserción, modificación y eliminación). El objetivo de separar el Modelo de acuerdo a los escenarios de evolución, es lograr una mejor comprensión y entendimiento del mismo, definiendo las trazas de cada escenario por separado.

La figura 70 muestra el modelo de trazabilidad para el escenario de evolución de inserción, considerando las etapas de Ingeniería de requisitos del Dominio y Diseño del Dominio, para la fase de Ingeniería del Dominio, y las etapas de Ingeniería de la Aplicación y Diseño de la Aplicación, para la fase de Ingeniería de la Aplicación.

La figura 71 muestra el modelo de trazabilidad para el escenario de evolución de inserción, considerando las etapas de Diseño del dominio e Implementación del Dominio, para la fase de Ingeniería del Dominio, y las etapas de Diseño de la Aplicación e Implementación de la Aplicación, para la fase de Ingeniería de la Aplicación.

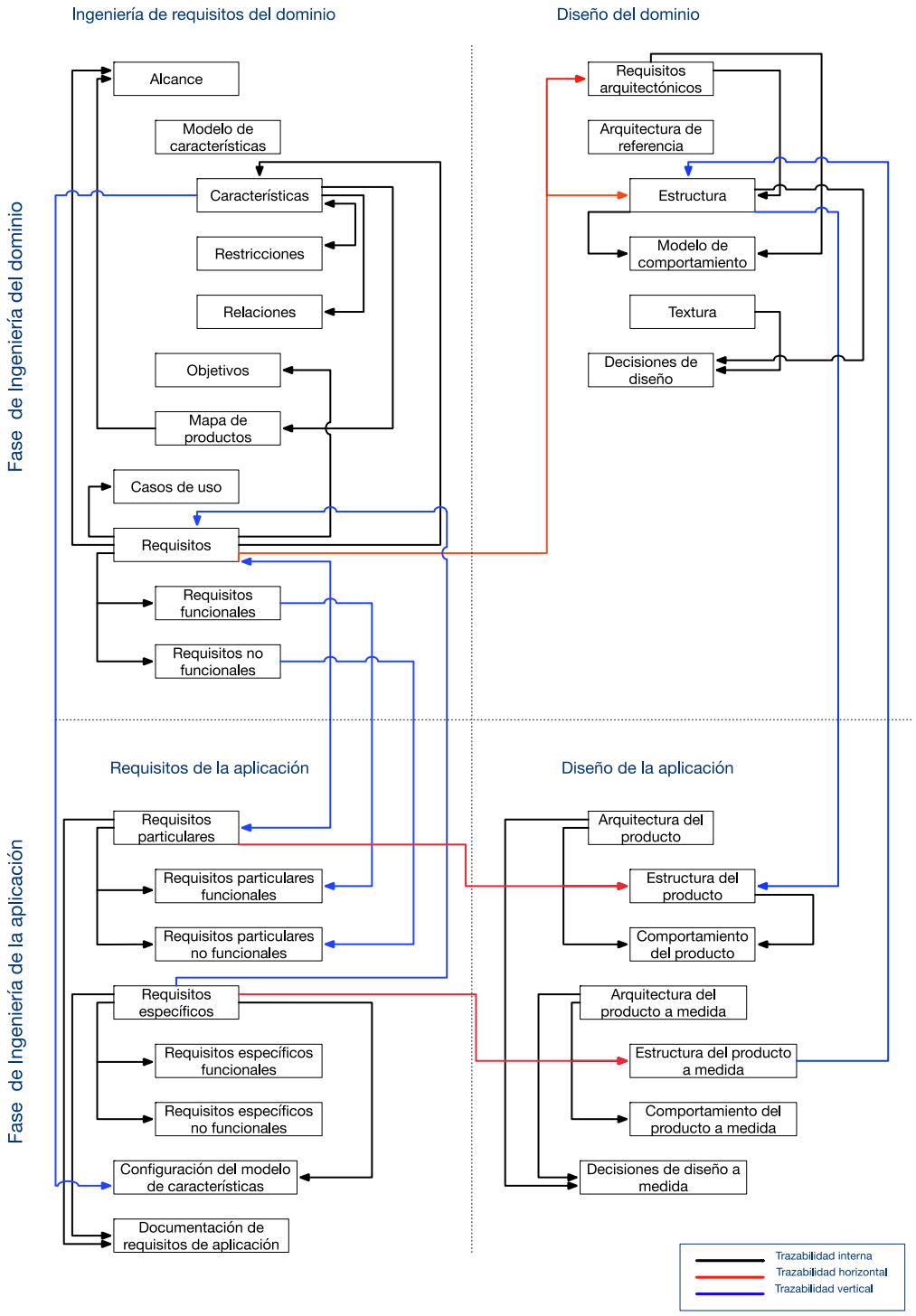

Figura 70: Modelo de trazabilidad para el escenario de evolución de inserción

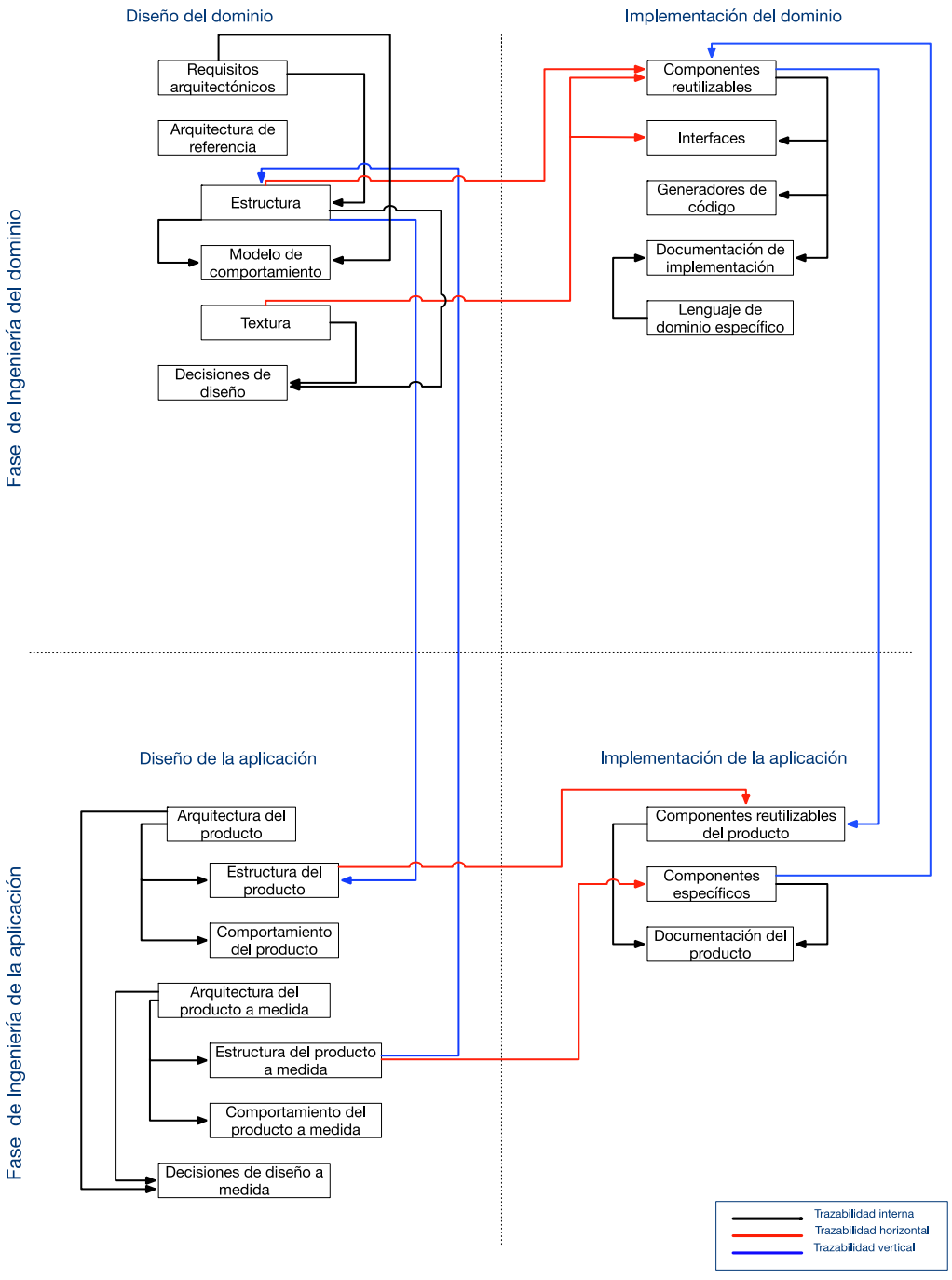

Figura 71: Modelo de trazabilidad para el escenario de evolución de inserción

La Figura 72 muestra el modelo de trazabilidad para el escenario de evolución de modificación, considerando las etapas de Ingeniería de requisitos del Dominio y Diseño del Dominio, para la fase de Ingeniería del Dominio, y las etapas de Ingeniería de la Aplicación y Diseño de la Aplicación, para la fase de Ingeniería de la Aplicación.

La Figura 73 muestra el modelo de trazabilidad para el escenario de evolución de modificación, considerando las etapas de Diseño del Dominio e implementación del Dominio, para la fase de Ingeniería del Dominio, y las etapas: Diseño de la Aplicación e Implementación de la Aplicación, para la fase de Ingeniería de la Aplicación.

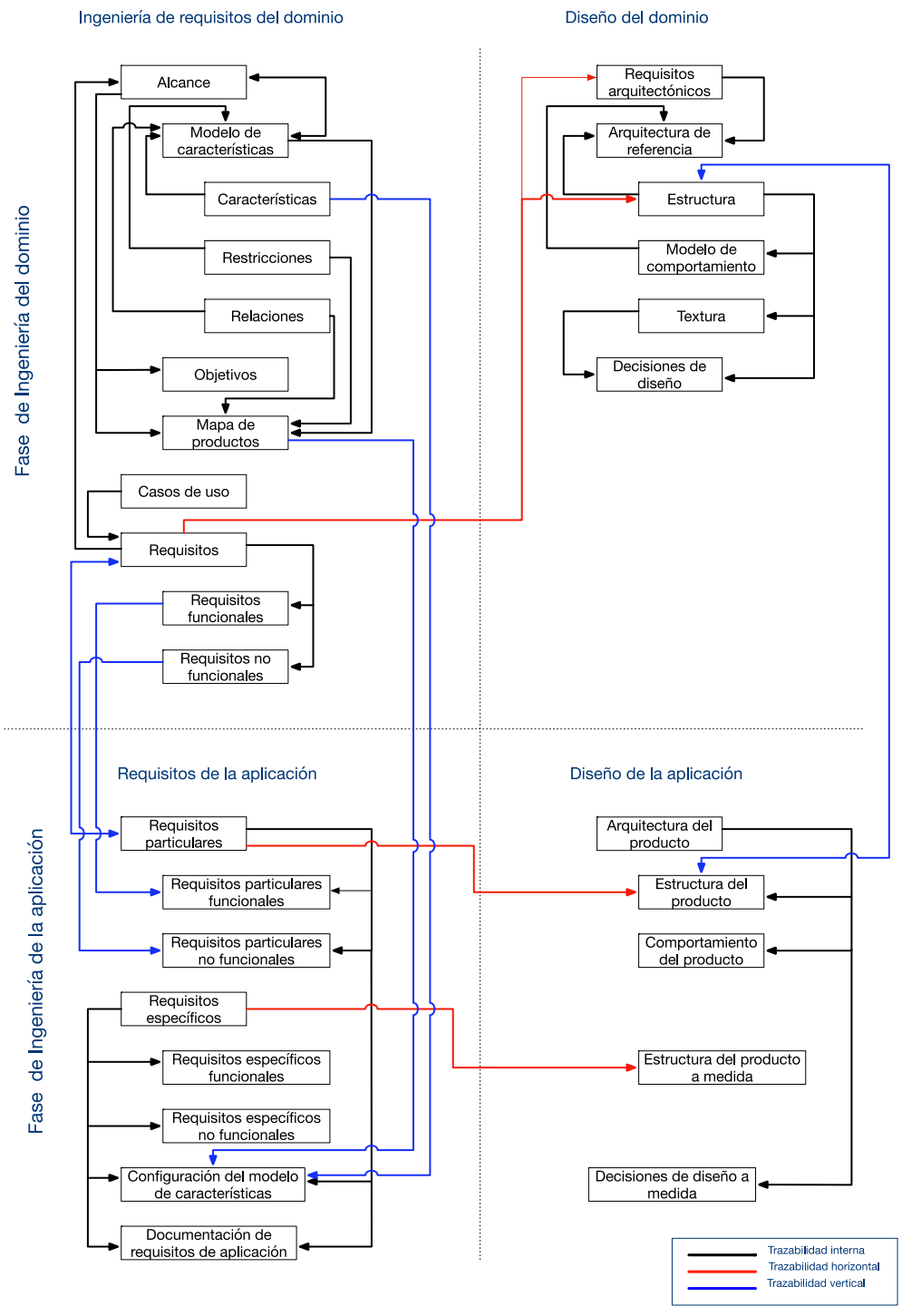

Figura 72: Modelo de trazabilidad para el escenario de evolución de modificación

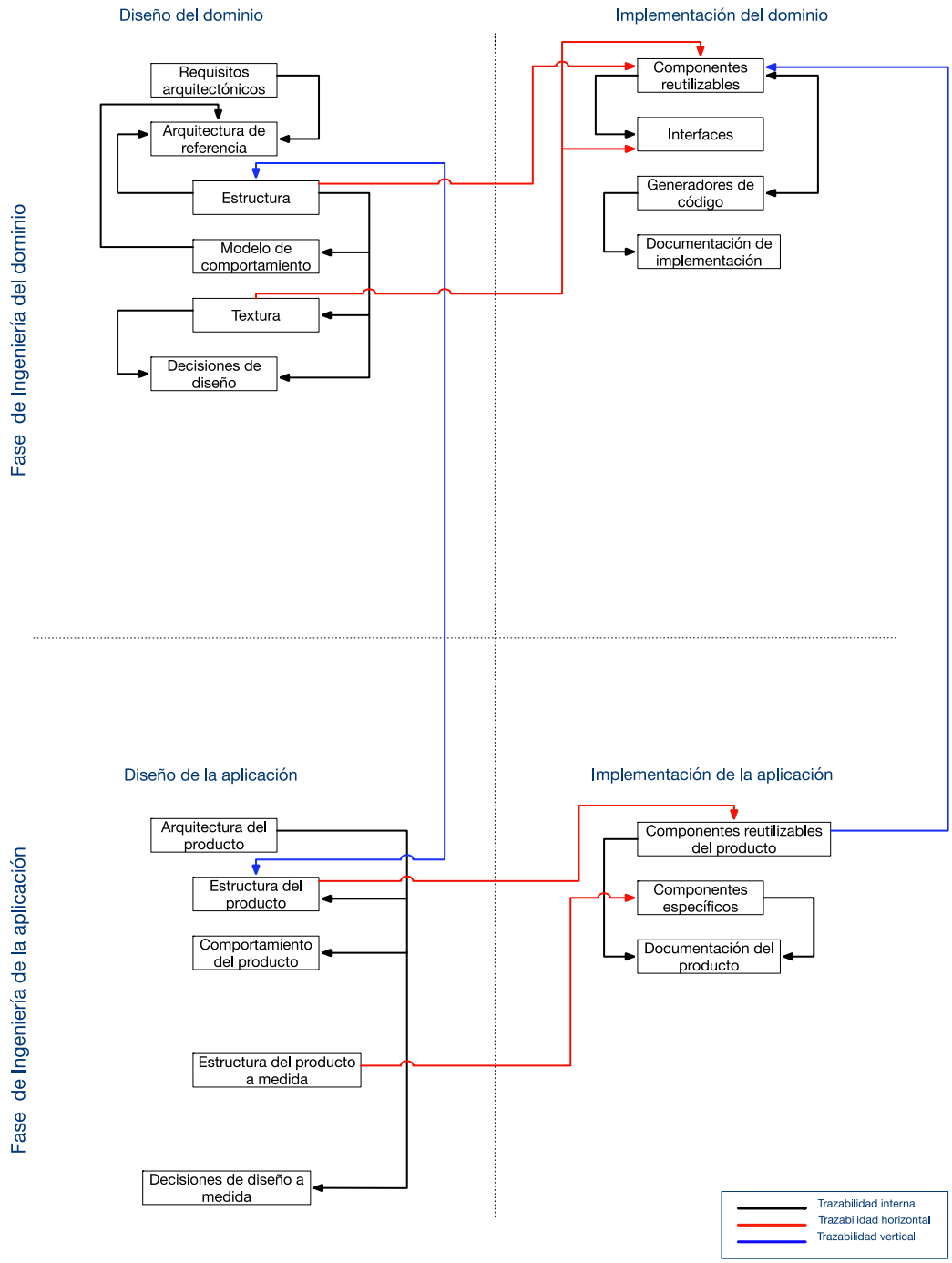

Figura 73: Modelo de trazabilidad para el escenario de evolución de modificación

La Figura 74 muestra el modelo de trazabilidad para el escenario de evolución de eliminación, considerando las etapas: Ingeniería de requisitos del Dominio y Diseño del Dominio, para la fase de Ingeniería del Dominio, y las etapas de Ingeniería de la Aplicación y Diseño de la Aplicación, para la fase de Ingeniería de la Aplicación.

La Figura 75 muestra el modelo de trazabilidad para el escenario de evolución de eliminación, considerando las etapas de Diseño del Dominio e Implementación del Dominio, para la fase de Ingeniería del Dominio, y las etapas de Diseño de la Aplicación e Implementación de la Aplicación, para la fase de Ingeniería de la Aplicación.

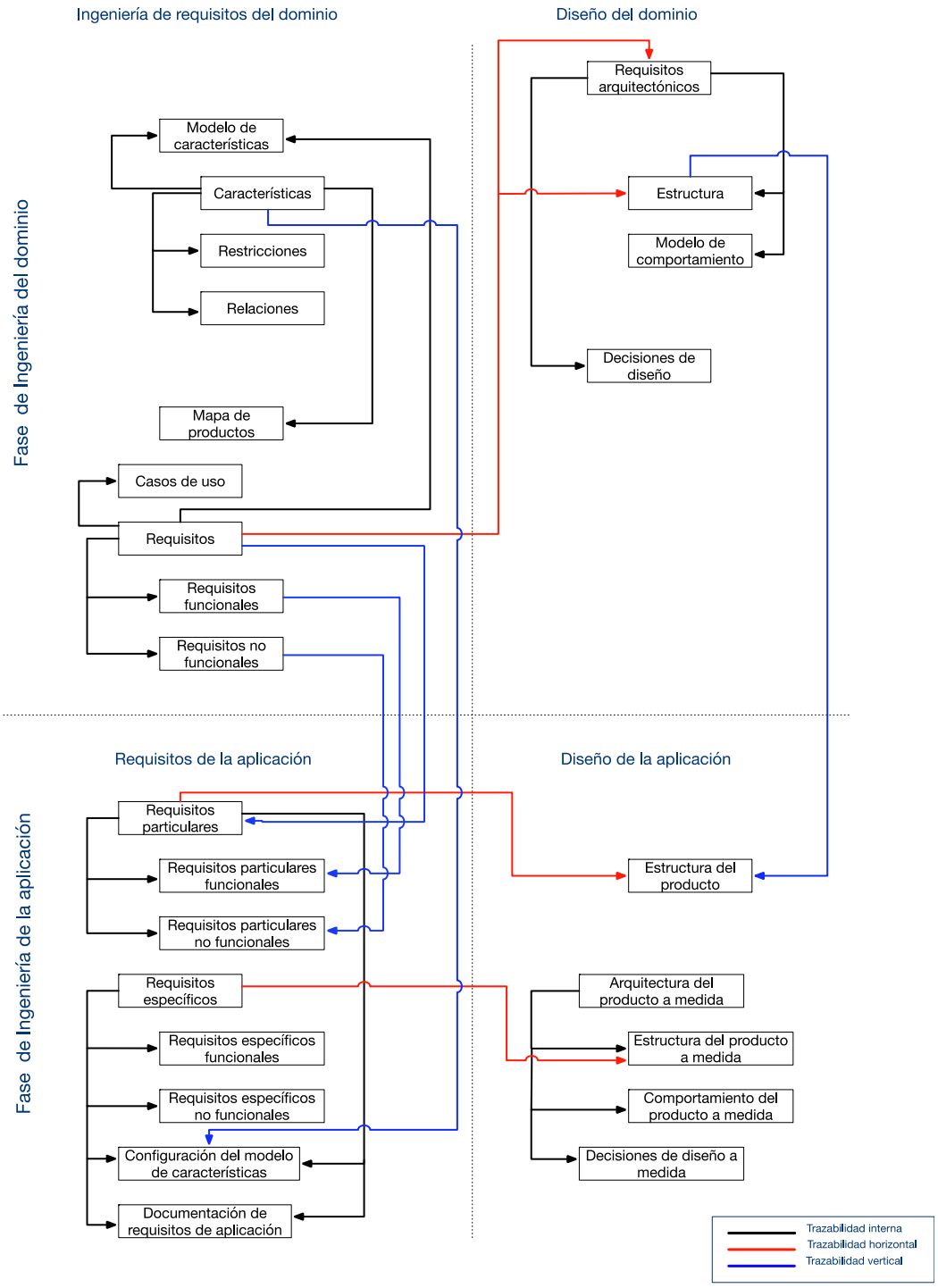

Figura 74: Modelo de trazabilidad para el escenario de evolución de eliminación

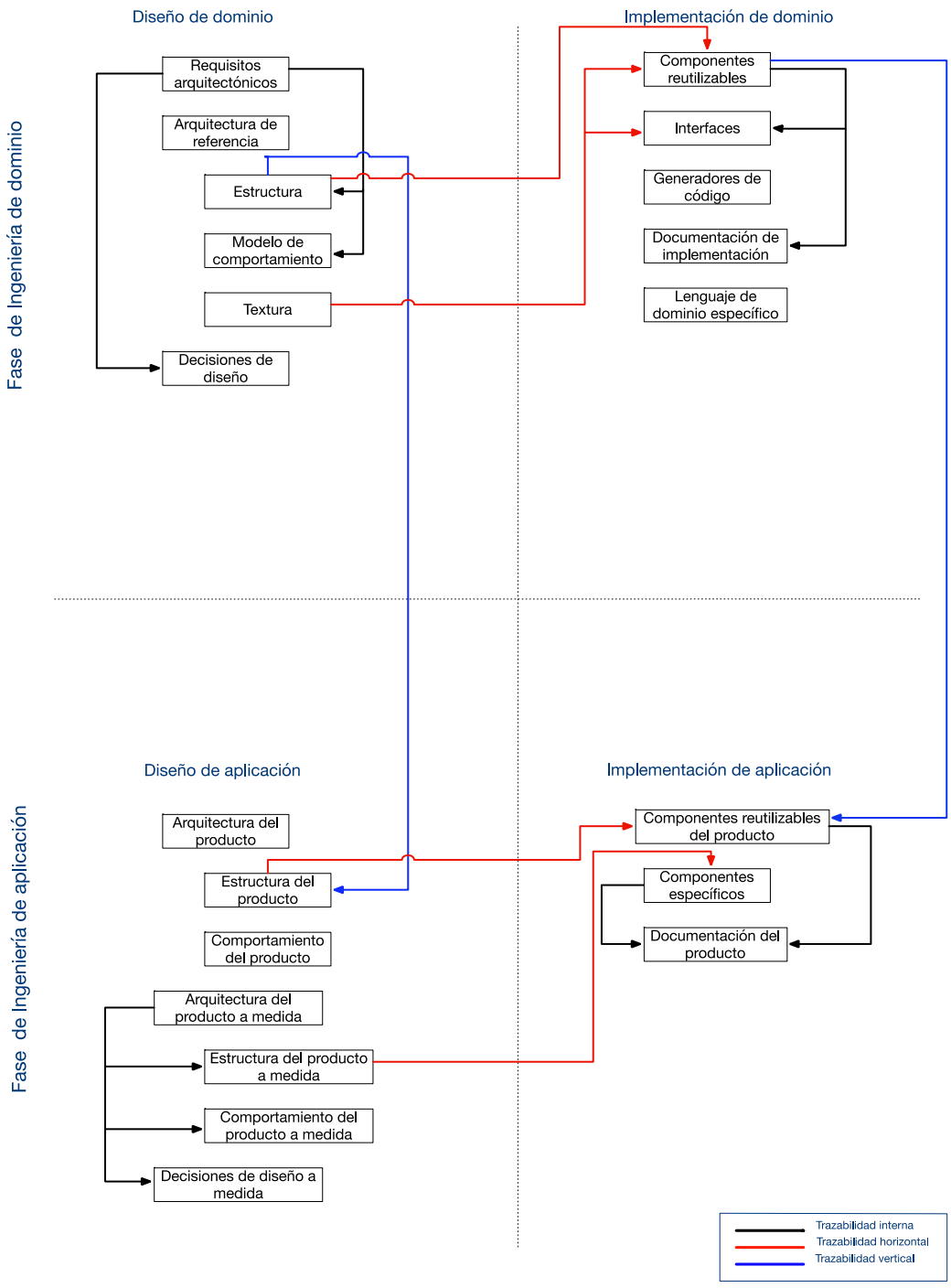

Figura 75: Modelo de trazabilidad para el escenario de evolución de eliminación

A continuación, se muestra el modelo de trazabilidad interna y trazabilidad vertical, para las etapas de Pruebas del dominio y Pruebas de la aplicación, considerando los tres

escenarios de evolución definidos en este proyecto de título (inserción, modificación y eliminación).

La figura 76 representa el modelo de trazabilidad interna y trazabilidad vertical entre las etapas Pruebas del Dominio y Pruebas de la Aplicación, para el escenario de evolución de inserción.

La Figura 77 representa el modelo de trazabilidad interna y trazabilidad vertical entre las etapas Pruebas del Dominio y Pruebas de la Aplicación, para el escenario de evolución de modificación.

La Figura 78 representa el modelo de trazabilidad interna y trazabilidad vertical entre las etapas Pruebas del Dominio y Pruebas de la Aplicación, para el escenario de evolución de eliminación.

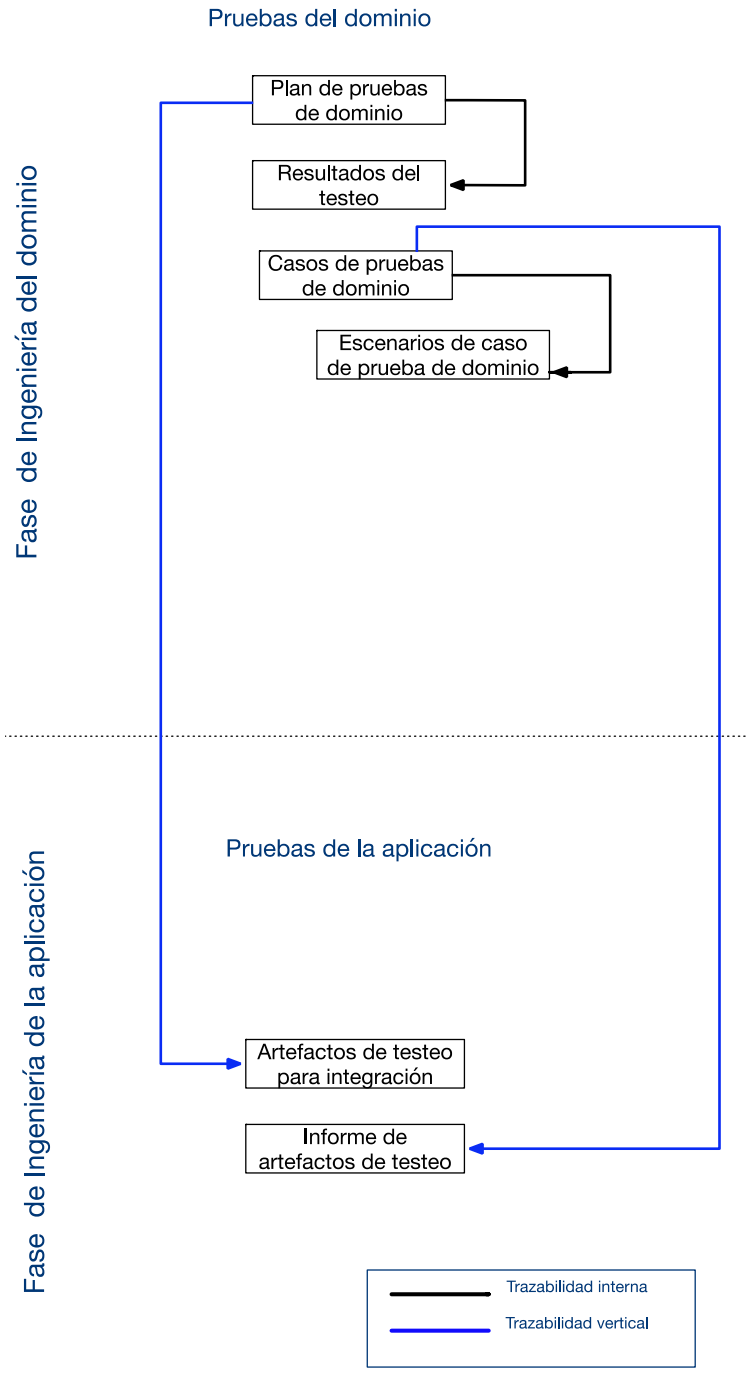

Figura 76: Modelo de trazabilidad en las etapas de Pruebas del Dominio y Pruebas de la Aplicación, para el escenario de evolución de inserción

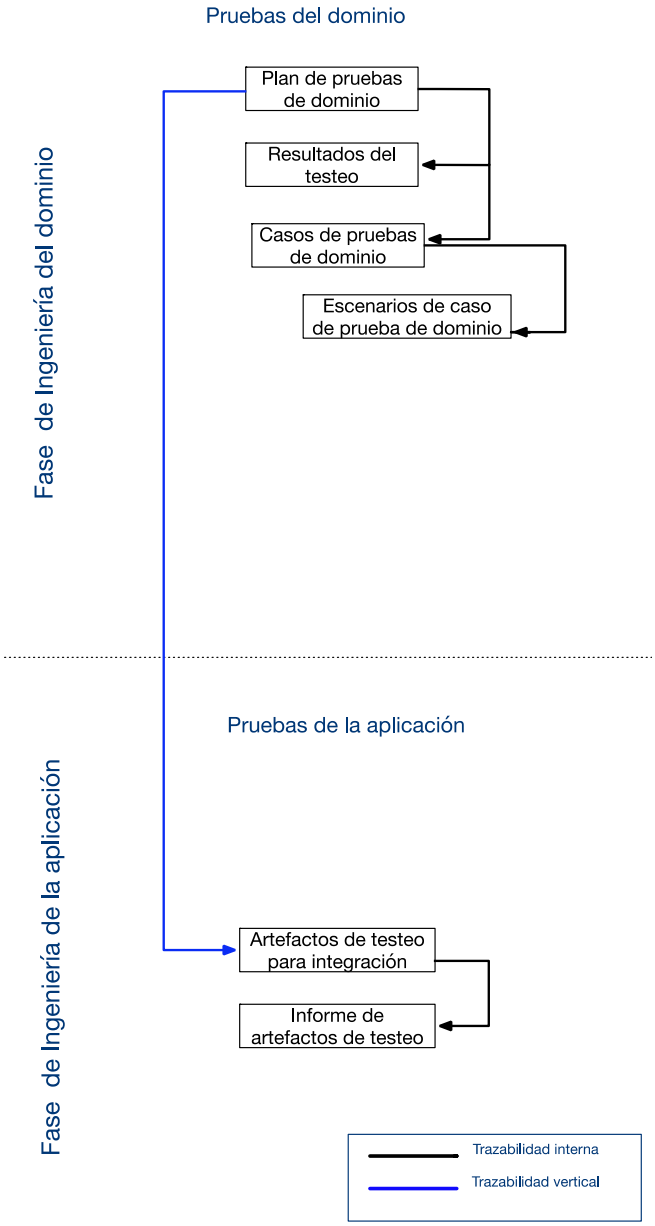

Figura 77: Modelo de trazabilidad entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de modificación

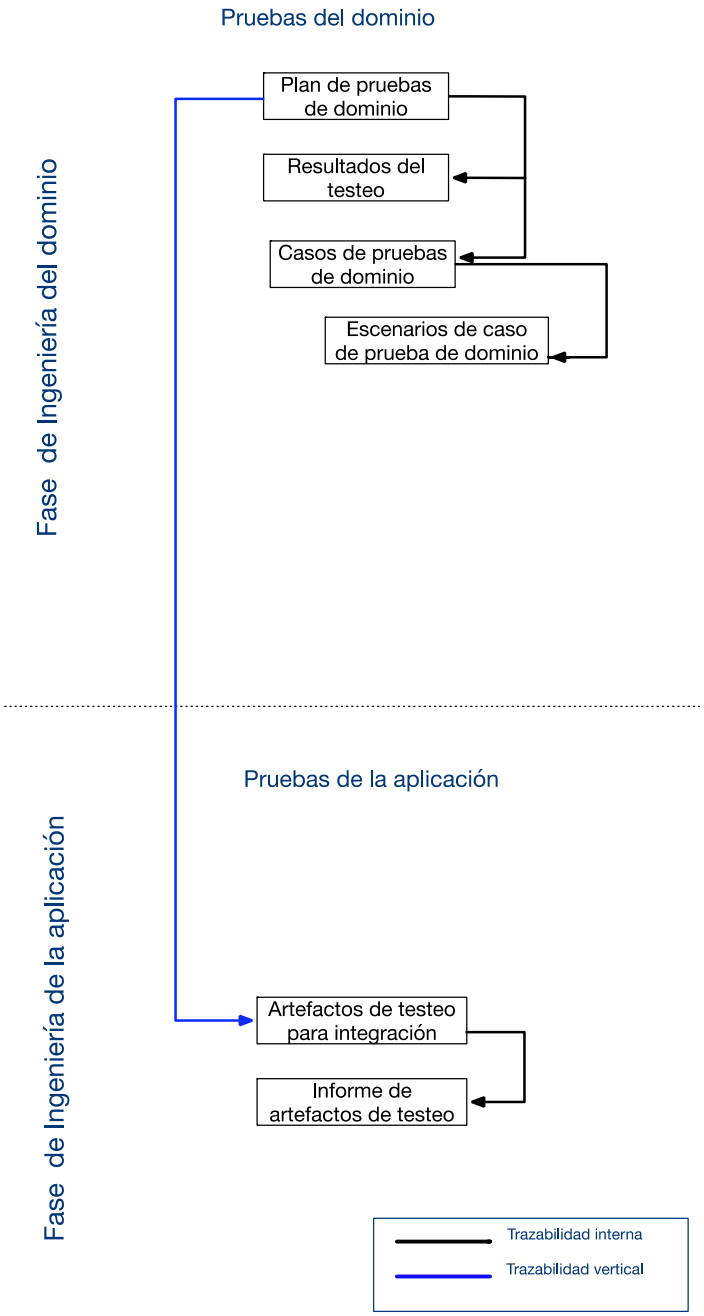

Figura 78: Modelo de trazabilidad entre las etapas de Pruebas del dominio y Pruebas de la aplicación, para el escenario de evolución de eliminación.

6. Roles asociados a los artefactos definidos para el modelo de trazabilidad

Cada rol está asociado a uno o más artefactos, definidos por el modelo de trazabilidad. El objetivo de esto es tener una comunicación más efectiva dentro del equipo de desarrollo de LPS ante una evolución de la línea de productos. Según Van der Linden, Schmid and Rommes (2007) se definen los roles para cada artefacto como sigue:

6.1. Ingeniería de requisitos del dominio

- Alcance: Gerente de productos; persona a cargo de la planificación y evolución de la gama completa de productos, tanto para los productos actuales y futuros de la línea de productos de software.
- Modelo de características: Gerente de producto; implica la planificación de los productos actuales y futuros de la línea de productos, sus características y su valor comercial. Está a cargo además de los artefactos que componen el modelo de características (características, relaciones y restricciones).
- Caso de uso: Ingeniero de requisitos del dominio; persona a cargo del desarrollo y mantención de los requisitos, además de los distintos escenarios de caso de uso que se presenten.
- Requisitos: Ingeniero de requisitos del dominio; trata con el desarrollo y mantención de todos los requisitos para la gama completa de productos. Está a cargo además de los requisitos funcionales y no funcionales del sistema.

6.2. Diseño del dominio

- Requisitos arquitectónicos: Arquitecto del dominio; es la persona que toma los requisitos más relevantes para ser diseñados en la arquitectura.

- Arquitectura de referencia: Arquitecto del dominio; es el encargado del desarrollo y mantención de la arquitectura de referencia para la gama completa de los productos. Además, estará a cargo de los artefactos que componen la arquitectura de referencia (estructura, modelo de comportamiento y textura).
- Decisiones de diseño: Arquitecto del dominio; toma todas las decisiones de la variabilidad del diseño, además de validar si los diseños de activos cumplen con la arquitectura de referencia.

6.3. Implementación del dominio

- Componente reutilizable: Desarrollador del dominio; es el encargado del desarrollo y mantención de los componentes reutilizables las interfaces para toda la gama de productos.
- Interfaces: Desarrollador del domino; su rol abarca el desarrollo y mantención de los componentes reutilizables y las interfaces para toda la gama de productos.
- Generador de códigos: Desarrollador del dominio; su rol abarca el desarrollo y mantención de los componentes reutilizables y las interfaces para toda la gama de productos.
- Lenguaje del dominio específico: Desarrollador del dominio; es aquel que desarrolla los componentes reutilizables.
- Documentación de implementación: Gestor de activos del dominio; es el responsable de la mantención de las versiones y las variantes de todos los activos de dominio de la gama completa de productos, además de realizar la documentación.

6.4. Pruebas del dominio

- Plan de pruebas del domino: Tester del dominio; realiza las pruebas de productos integrados, la interpretación y las pruebas de sistema en los activos del dominio.

- Resultados del Testeo: Tester del dominio; realiza las pruebas de productos integrados, la interpretación y las pruebas de sistema en los activos de dominio.
- Casos de prueba del dominio: Tester del dominio; realiza las pruebas de productos integrados, la interpretación y las pruebas del sistema en los activos del dominio. Estará a cargo además de los escenarios de caso de pruebas del domino.

6.5. Requisitos de la aplicación

- Requisitos particulares: Ingeniero de requisitos de la aplicación; este rol abarca el desarrollo y mantención de los requisitos del producto, además este proporciona la retroalimentación para el gestor de productos en la viabilidad y el costo de la selección de características específicas. Está a cargo además de los artefactos que componen los requisitos particulares (requisitos particulares funcionales y no funcionales).
- Requisitos específicos: Ingeniero de requisitos de la aplicación; a cargo del desarrollo y mantención de los requisitos de un producto en singular, además de satisfacer los requisitos en específicos del producto. Está a cargo además de los artefactos que componen los requisitos específicos (Requisitos específicos funcionales y no funcionales).
- Configuración del modelo de características: Ingeniero de requisitos de la aplicación; es el que abarca el desarrollo y mantención de los requisitos de un producto en singular.
- Documentación de requisitos de la aplicación: Ingeniero de requisitos de la aplicación; tanto los requisitos instanciados, como los creados, deben ser documentados por el Ingeniero de requisitos de la aplicación.

6.6. Diseño de la aplicación

- Arquitectura del producto: Arquitecto de la aplicación; es el encargado del desarrollo y mantención de la arquitectura para un producto en singular. Está a cargo además de los artefactos que componen la Arquitectura del producto (estructura del producto y comportamiento del producto).
- Arquitectura del producto a medida: Arquitecto de la aplicación; es el encargado del desarrollo y mantención de la arquitectura para un producto en singular. Está a cargo además de los artefactos que componen la arquitectura del producto a medida (estructura del producto a medida y comportamiento del producto a medida).
- Decisiones de Diseño del producto: Arquitecto de la aplicación; es el encargado del desarrollo y mantención de la arquitectura para un producto en singular.

6.7. Implementación de la aplicación

- Componentes reutilizables del producto: Desarrollador de la aplicación; es el encargado del desarrollo y mantención de componentes e interfaces específicas de la aplicación, implicando la reutilización de componentes e interfaces de dominio y su especialización hacia la aplicación.
- Componentes específicos: Desarrollador de la aplicación; es el encargado del desarrollo y mantención de componentes e interfaces específicas de la aplicación, implicando la reutilización de componentes e interfaces de dominio y su especialización hacia la aplicación.
- Documentación del producto: Desarrollador de la aplicación; encargado de documentar las decisiones que se tomaron para crear el producto en particular, junto con los componentes instanciados y específicos de un producto en particular.

6.8. Pruebas de la aplicación

- Informe de problemas: Tester de la aplicación; es el encargado de realizar pruebas de las aplicaciones singulares y comunicarlas a la etapa correspondiente.
- Artefactos de testeo para integración: Tester de la aplicación; es el encargado de realizar pruebas de las aplicaciones singulares y comunicarlas a la etapa correspondiente.
- Informe de artefactos de testeo: Tester de la aplicación; es el encargado de realizar pruebas de las aplicaciones singulares y comunicarlas a la etapa correspondiente.

Se muestra a continuación el modelo de artefactos de trazabilidad, con sus respectivos roles asociados. Los roles definidos en el modelo, se encargarán del artefacto correspondiente al color que les corresponda. Este modelo es válido para los tres escenarios de evolución definidos en este proyecto. Ver figura 79.

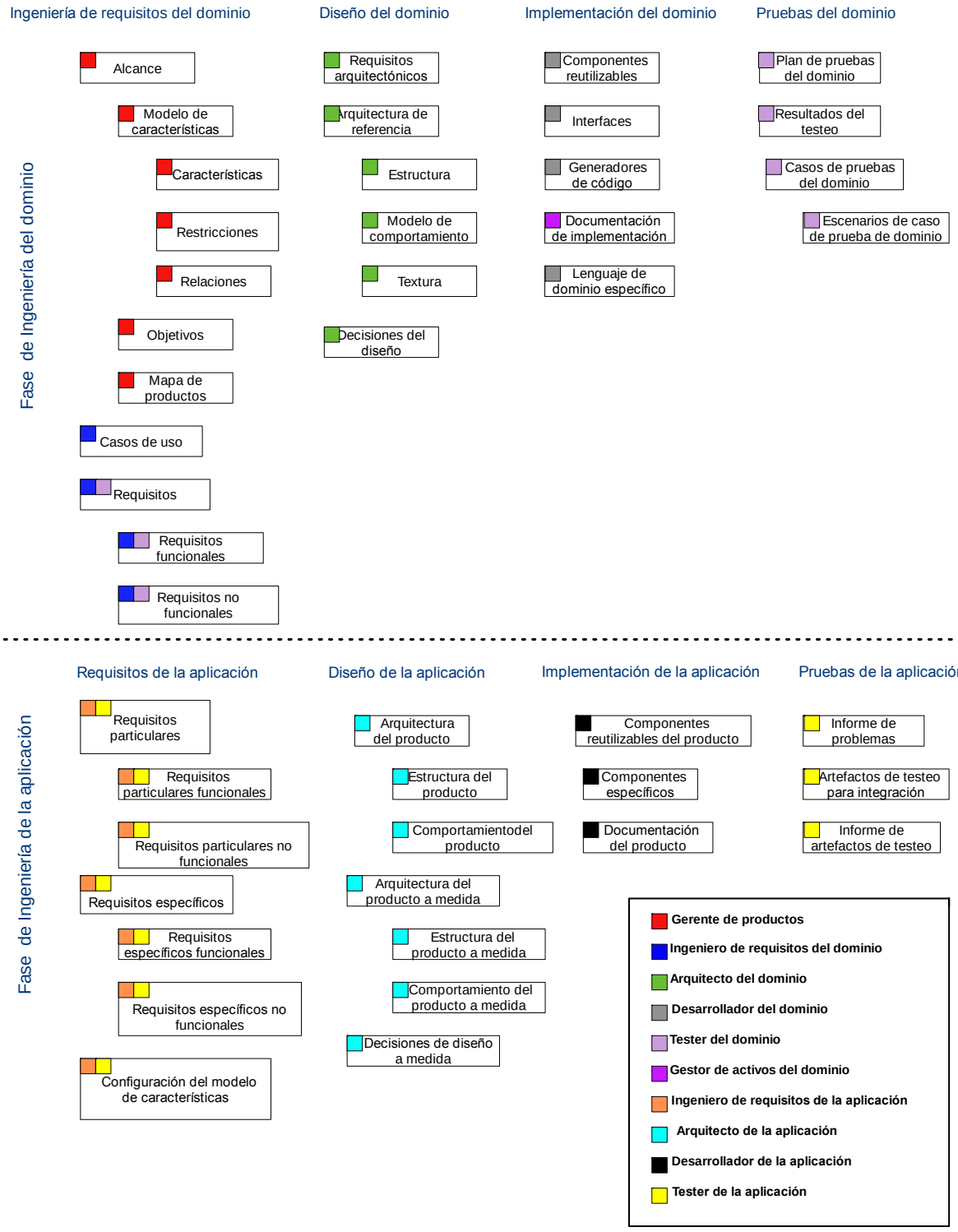

Figura 79: Modelo de Roles y artefactos de trazabilidad

7. Caso de estudio

Para la validación del Modelo de trazabilidad, se utilizó un caso de estudio desarrollado en el proyecto de título que lleva por nombre "Mantención de una familia de productos en el dominio de administración de emergencias", realizado por Ana Montero Cáceres, alumna de la Universidad Católica de la Santísima Concepción de la carrera Ingeniería Civil Informática, en el año 2016, titulada en el año 2017. El objetivo principal de su proyecto consistió en expandir la familia de productos existentes en el dominio de administración de emergencias, por medio del desarrollo de componentes reutilizables a partir de la aplicación móvil *CollabMap*. Esta aplicación está diseñada para colaborar conjuntamente con el Cuerpo de Bomberos ante una situación de emergencia, específicamente en incendios, con la finalidad de facilitar el procedimiento de una emergencia desde que ésta es recibida, hasta que ha finalizado [33].

Cabe destacar que en el proyecto de Ana Montero se hicieron dos actualizaciones en el sistema, las cuales corresponden dentro del Modelo de trazabilidad, al escenario de evolución de inserción. Por lo tanto, se analizará solo uno de los tres escenarios de evolución (inserción, modificación y eliminación) que contempla el Modelo de trazabilidad.

El funcionamiento de la aplicación consiste en el despliegue de un listado con todas las emergencias ingresadas previamente a una base de datos, la cual es llenada a través de una aplicación web controlada por la central de emergencias. En este listado de emergencias se muestran: el nombre de la emergencia, la distancia aproximada respecto a la ubicación actual del usuario, el tiempo de inicio del incendio y la cantidad de carros bomba que asisten a la emergencia, como se muestra en la figura 80.

Todas las figuras mostradas en este capítulo fueron obtenidas del proyecto de título de Ana Montero [33].

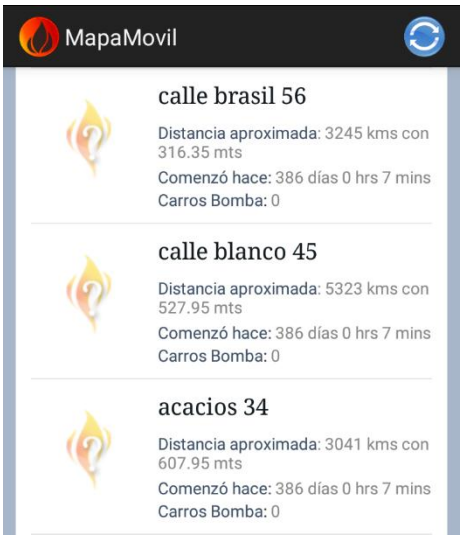

Figura 80: Listado de emergencias e información asociada

Luego, al seleccionar una emergencia, se desplegará un menú de opciones con las siguientes funcionalidades: Asistir esta emergencia, Mostrar ubicación, Mostrar ruta hacia la emergencia, Recursos que la asisten, Recursos digitales asociados y Crear un recurso digital, tal como se muestra en la figura 81.

Figura 81: Menú de opciones asociado a una emergencia seleccionado del listado principal

A través del tiempo se han ido realizando modificaciones de acuerdo a las exigencias actuales de bomberos y a las actualizaciones en las APIs de Android [33]. En este caso de estudio se agregaron dos nuevos componentes reutilizables para aportar al desarrollo de esta familia de productos: *Creación, envío y recepción del recurso digital de audio* y *Visualización de ruta hacia la emergenci*a, los cuales serán explicados a continuación [33].

- Creación, envío y recepción del recurso digital de audio: "Esta funcionalidad permite a un usuario grabar un audio asociado a una emergencia en específico y guardarlo tanto en el repositorio de datos local, como en la base de datos externa, para luego dar la opción al usuario de compartir dicho recurso con usuarios que estén conectado en la misma emergencia.

  Cabe mencionar que parte de esta funcionalidad fue una modificación a la ya implementada por Erika Ormeño [34], de tal forma de añadir la generación de audio a los recursos digitales existentes".

Para llevar a cabo esta funcionalidad se crearon/modificaron los siguientes componentes:

- ConcreteAudioMakerA: Este componente es el encargado de crear el audio.
- AudioMaker: Este componente corresponde a la interfaz implementada para la clase ConcreteAudioMakerA.
- DigitalResourcesMaker: Este componente permite al usuario seleccionar qué recurso digital (fotografía, video o audio) desea crear, siendo cada uno de ellos implementado por su componente respectivo.
- ConcreteOptionalComponentsA: Este componente está encargado de administrar los componentes opcionales presentes en la aplicación.
- DefaultOptionalComponents: Este componente permite gestionar las instancias de aquellos componentes opcionales e independientes del sistema operativo Android.
- OptionalComponents: Este componente corresponde a la interfaz implementada para la clase "ConcreteOptionalComponentsA".

- ConcreteAudioOpenerA: Este componente permite gestionar la apertura de un audio.
- AudioOpener: Este componente corresponde a la interfaz implementada para la clase “ConcreteAudioOpenerA”.
- ConcreteDigitalResourcesOpener: Este componente permite gestionar la apertura de los tres recursos digitales presentes en el proyecto, esto es, fotografía, videos y audio.
- DigitalResourceTypes: Este componente permite definir los formatos y los tipos de recursos digitales presentes formatos y los tipos de recursos digitales y que son gestionados por el componente “ConcreteDigitalResourceTypes”.

Los componentes de la lista anterior que fueron modificados para llevar a cabo esta funcionalidad fueron:

- “DigitalResourcesMaker”
- “ConcreteOptionalComponentsA”
- “DefaultOptionalComponents”
- “OptionalComponents”
- “ConcreteDigitalResourcesOpener”
- “DigitalResourceTypes”

Los componentes de la lista anterior que fueron creados son para llevar a cabo esta funcionalidad fueron:

- “ConcreteAudioMakerA”
- “AudioMaker”
- “ConcreteAudioOpenerA”
- “AudioOpener”

La interfaz del menú para esta funcionalidad se muestra en la figura 82.

Figura 82: Interfaz del menú de la emergencia para la creación de audio

El segundo componente reutilizable añadido a esta familia de productos es:

- Visualización de ruta hacia la emergencia: "Esta funcionalidad permite a un usuario, que es partícipe de una emergencia, visualizar la ruta desde el punto en el que está ubicado hasta el lugar de la emergencia, pensado para aquellos bomberos que desconocen la ubicación exacta de la emergencia y minimizar así los tiempos de llegada.

  Cabe mencionar que parte de esta funcionalidad fue una modificación a la ya implementada [34], de tal forma de añadir la visualización de ruta hacia la emergencia".

Para llevar a cabo esta funcionalidad, se crearon/modificaron los siguientes componentes [33]:

- EmergencyMenu: Esta actividad está conformada por un menú de opciones existentes sobre una emergencia.
- CommonLabels: Es la encargada de administrar los rótulos que son usados por varios componentes
- RouteToEmergencyMap: Este componente es el encargado de trazar la ruta desde el origen, que es la ubicación del usuario, hasta el lugar de la emergencia.
- DirectionsJSONParser: Este componente es el encargado de conectar con el web service de Google Maps y recibir el resultado en formato JSON, que retorna un listado que contiene latitudes y longitudes para luego interpretar el resultado.

Los componentes modificados de la lista anterior para llevar a cabo esta funcionalidad fueron:

- "EmergencyMenu"
- "CommonLabels"

Los componentes de la lista creados para llevar a cabo esta funcionalidad fueron:

- "RouteToEmergencyMap"
- "DirectionsJSONParser"

La interfaz del menú para esta funcionalidad se muestra en la figura 83.

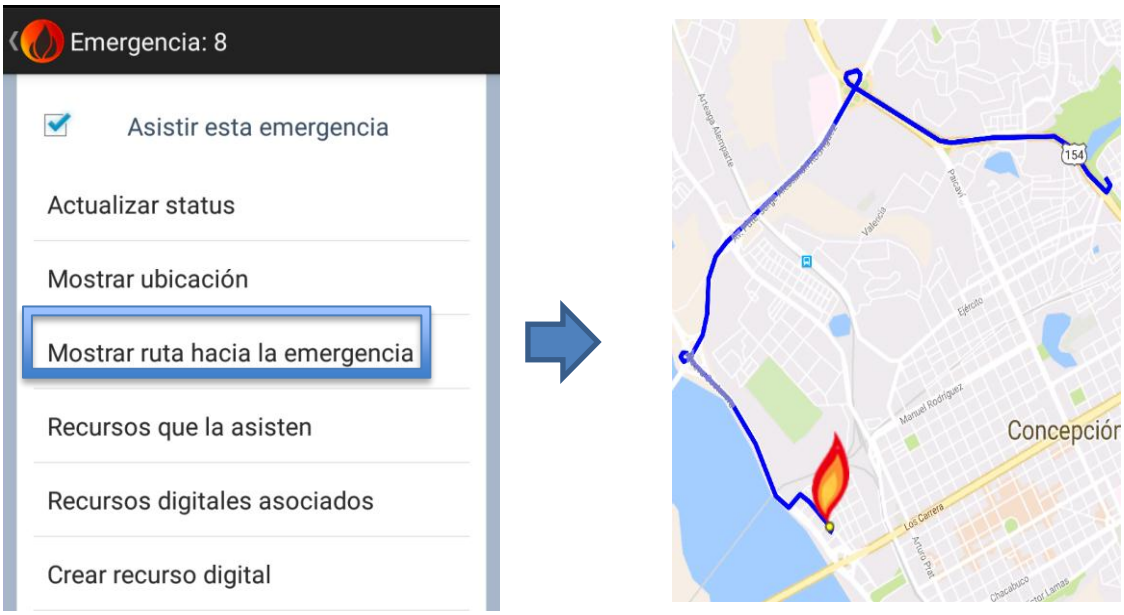

Figura 83: Interfaz del menú para mostrar la Ruta hacia la emergencia

7.1. Análisis del caso de estudio

En la fase de Ingeniería del dominio se agregó una nueva característica al modelo de características, ésta es: *Creación, envío y recepción del recurso digital de audio*. Además, se implementó una característica que ya estaba definida previamente en el modelo de características, ésta es: *Visualización de ruta hacia la emergencia*. Estas características surgieron de los requisitos propuestos por las partes interesadas, de acuerdo a la necesidad de acortar tiempos de llegada a la emergencia y para poder enviar la información de manera más rápida de la que se podría a través de un teclado. En la figura 84 se resalta en rojo la característica del modelo de características que fue implementada y en la figura 85 se resalta en rojo la característica que se agregó al modelo de características. Ambas figuras han sido recortadas y modificadas desde el modelo de características original, obtenido de [35], con el objetivo de lograr una mayor comprensión de lo implementado en el proyecto de Ana Montero.

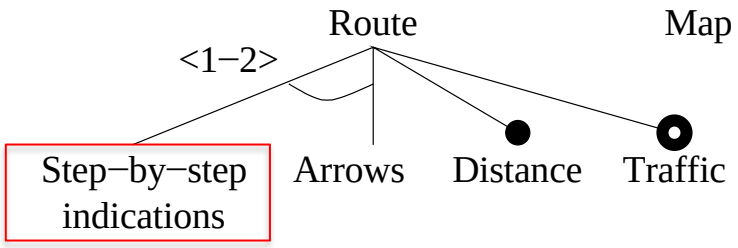

Figura 84: Característica implementada en el proyecto de Ana Montero

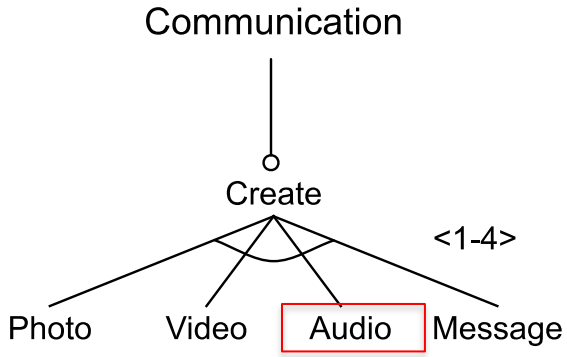

Figura 85: Característica creada en el proyecto de Ana Montero

Ya que se han definido ambas características, se procederá a evaluar la trazabilidad que se obtuvo del caso de estudio, respecto del modelo de trazabilidad propuesto en este proyecto de título, con el objetivo de validar si el modelo de trazabilidad cumple con el objetivo de apoyar una evolución de una línea de productos de software.

En la figura 86 se muestra la validación del modelo de trazabilidad, en las etapas de Ingeniería de Requisitos del dominio, Diseño del dominio, Requisitos de la aplicación y Diseño de la aplicación. Cabe destacar que se eliminaron de esta figura todos aquellos artefactos que no son relevantes para la validación del modelo, ya que no son contemplados

por el caso de estudio. Los artefactos eliminados son: Requisitos específicos, Requisitos específicos funcionales, Requisitos específicos no funcionales, Arquitectura del producto específico, Estructura del producto específico, Comportamiento del producto específico y la Textura.

En la Figura 87 se muestra la validación del modelo de trazabilidad, en las etapas de Diseño del dominio, Diseño de la aplicación, Implementación del dominio e Implementación de la aplicación. Cabe destacar que se eliminaron de esta figura todos aquellos artefactos que no son relevantes para la validación del modelo, ya que no son contemplados por el caso de estudio, estos son: Textura, Arquitectura del producto a medida, Estructura del producto a medida, Comportamiento del producto a medida, Requisitos arquitectónicos y Lenguaje de dominio específico.

Se representa con una flecha continua, la coincidencia entre el caso de estudio y el modelo de trazabilidad propuesto, en cuanto a la trazabilidad de los artefactos definidos por el modelo de trazabilidad. La no coincidencia entre el caso de estudio y el modelo de trazabilidad, en cuanto a los artefactos definidos por el modelo de trazabilidad, está representada por las flechas discontinuas.

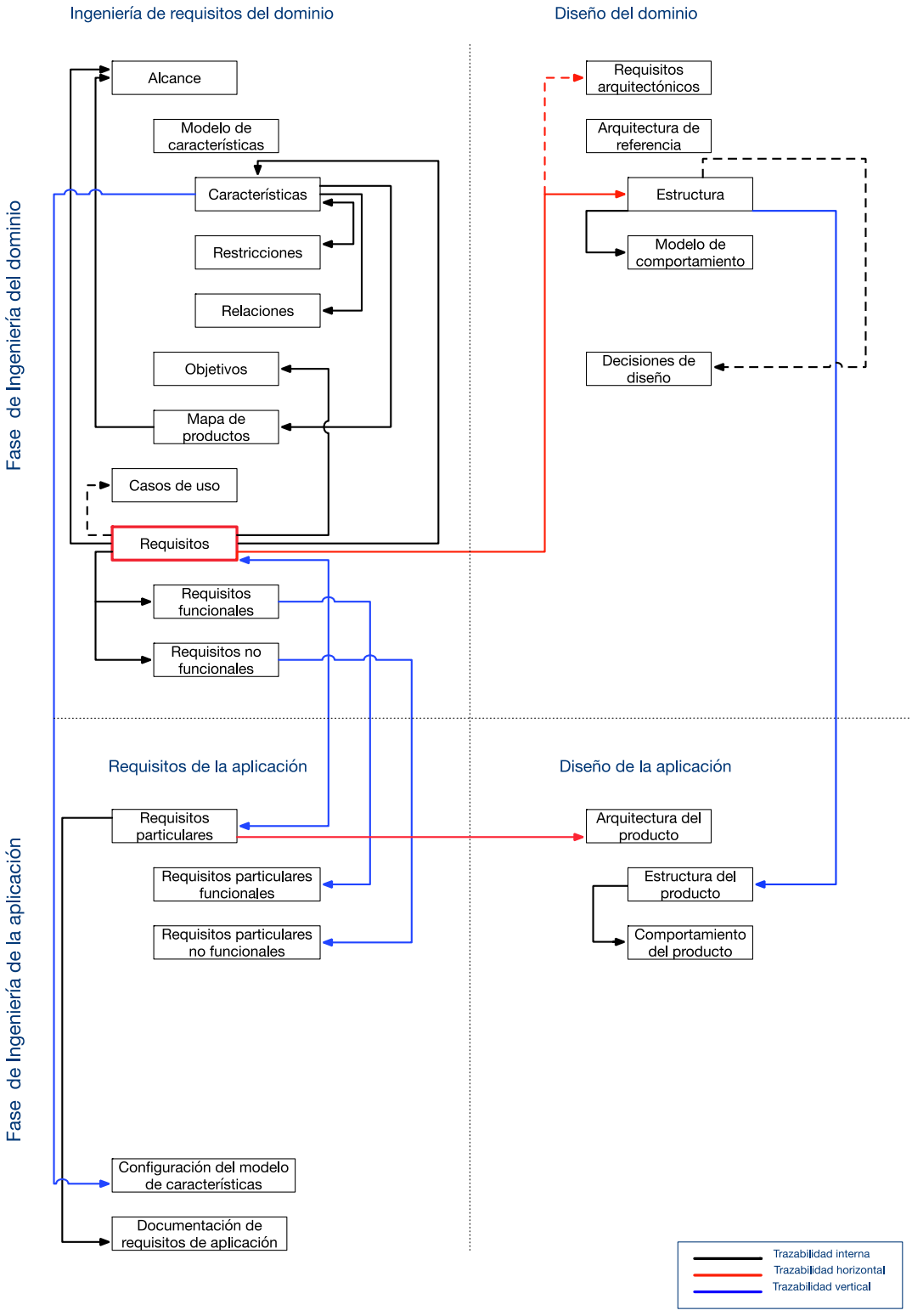

Figura 86: Validación del modelo de trazabilidad utilizando el caso de estudio

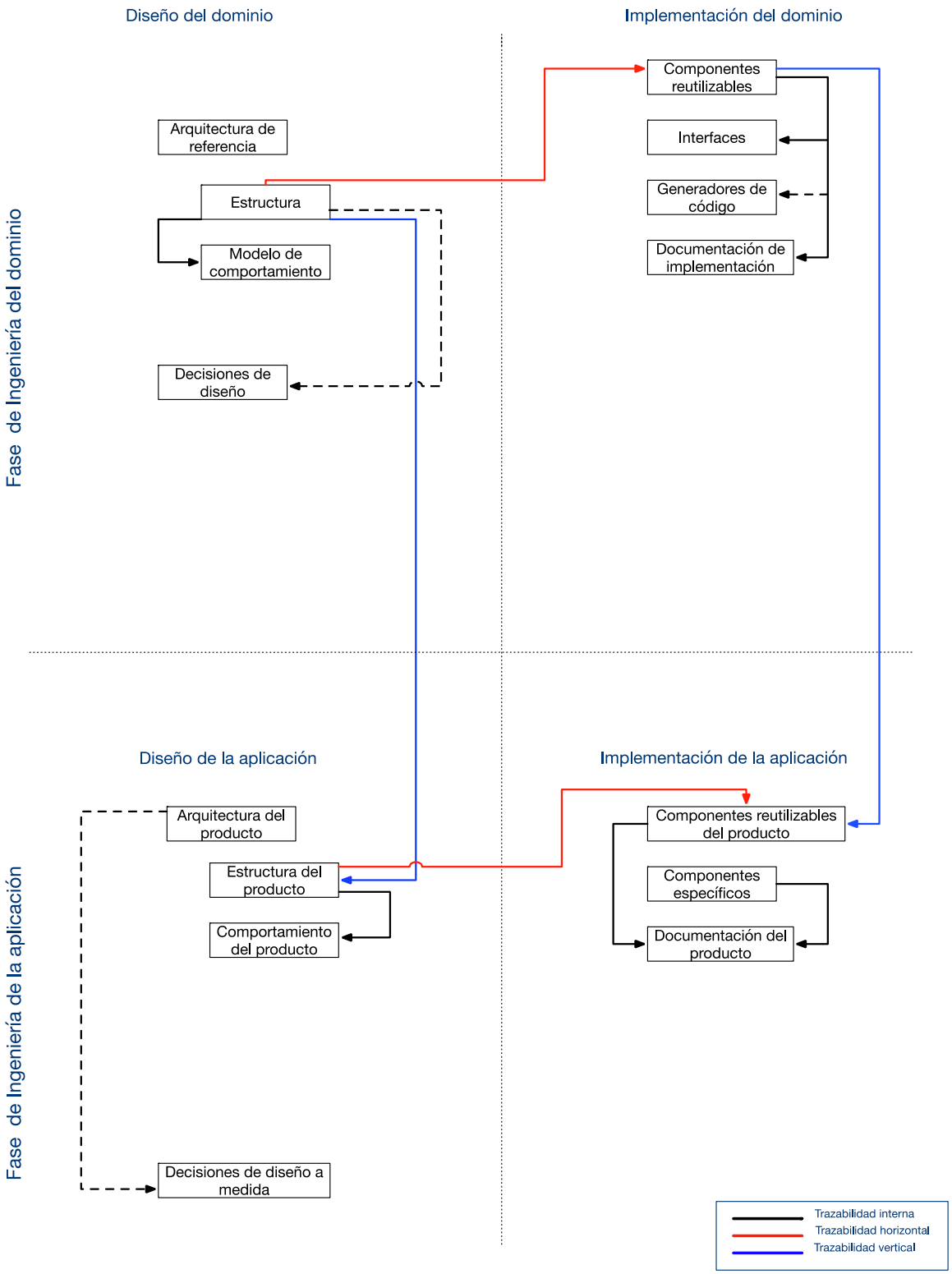

Figura 87: Validación del modelo de trazabilidad utilizando el caso de estudio

Cabe destacar que el análisis de trazabilidad, será casi el mismo para ambas características implementadas. La diferencia está en que la funcionalidad Visualización de la ruta hacia la emergencia ya estaba definida previamente como característica dentro de la LPS, por lo que la traza de los artefactos; características, relaciones y restricciones no deben ser considerados por el lector de este proyecto de título al momento de analizar esas trazas. Sin embargo, sí deben ser consideradas para la otra funcionalidad implementada (Creación, envío y recepción de audio).

Se destacó de color rojo en la figura 86 el artefacto Requisito, ya que es el artefacto creado por Ana Montero, que da origen a todas las demás trazas en el modelo de trazabilidad. Las demás trazas a partir del artefacto Requisito, serán explicadas a continuación [33].

Análisis para la trazabilidad interna en la etapa de Ingeniería de requisitos de dominio:

- De requisitos a requisitos funcionales: Efectivamente se crearon y documentaron en el proyecto de Ana Montero los Requisitos funcionales, tales como: *formato, fecha, quién envía el audio, ubicación, persona encargada de emergencia*, entre otros.
- De requisitos a requisitos no funcionales: Se crearon y documentaron en el proyecto de Ana los Requisitos no funcionales, tales como: *se necesitan permisos para utilizar el micrófono del dispositivo, se requiere un mínimo de almacenamiento*, entre otros.
- De requisitos al alcance: efectivamente se amplió el Alcance de la LPS, debido a que se amplió la familia de productos existentes en el dominio de administración de emergencias, tal como se define en el objetivo general del proyecto de Ana.
- De requisitos a características: el Requisito implementado, es seleccionada como característica visible para el usuario final del sistema, con sus respectivas relaciones y restricciones de comunicación, tal como se representó en la Figura 87.
- De requisitos a objetivos: el nuevo requisito modificó los objetivos, como se plantea en el modelo, ya que las partes interesadas tienen por objetivo lograr una mejor comunicación ante una emergencia de bombero.

- De requisitos al caso de uso: esta traza en particular no se cumplió como se esperaba para el modelo de trazabilidad. Ana comentó que no fue necesario incorporar un caso de estudio, ya que no era necesario ilustrar el comportamiento del requisito ante un evento en el sistema [6].

Análisis para la trazabilidad horizontal entre las etapas de Ingeniería de requisitos de dominio y Diseño de dominio:

- De requisitos a requisitos arquitectónicos: No fue considerado como requisito arquitectónico, ya que éstos se describen como los primeros requisitos que forman parte de la Arquitectura de referencia, debido a que cuando comienza el desarrollo de la LPS, no todos los requisitos alcanzan a estar definidos a tiempo [20]. Para el caso de estudio, el nuevo requisito de pensó luego de que las partes interesadas se percataran que las funcionalidades de comunicación existentes no satisfacían del todo a la LPS.
- De requisitos a estructura: efectivamente se crearon los componentes de la estructura, a partir del nuevo requisito. Un ejemplo de estos componentes diseñados en la etapa de diseño es: *AudioMaker* y *ConcreteAudioMakerA*, diseñados para gestionar la creación de audio en la interfaz de la aplicación y la creación de audio dentro de la aplicación, respectivamente. Se definió, además, el comportamiento de los componentes de la estructura, definiendo, por ejemplo, que el componente *AudioMaker* depende del componente *ConcreteAudioMakerA* para funcionar.

En el proyecto de Ana Montero, no se consideró necesario documentar las decisiones de diseño que se tomaron para incorporar o desechar ciertos componentes de diseño, para futuras modificaciones, debido a que no se pensó en otras modificaciones a futuro.

[6] Ana Montero. Conversación personal. Validación del modelo de trazabilidad.

Análisis para la trazabilidad horizontal entre las etapas de Diseño del dominio e Implementación del Dominio:

- De la estructura a componentes reutilizables: la estructura, la cual se compone de los distintos componentes de diseño y sus relaciones [20], se implementó como un componente reutilizable, es decir, se creó el módulo de código fuente en base a los componentes de diseño definidos para las características implementadas en el proyecto de Ana Montero. Estos componentes fueron explicados en el capítulo Caso de estudio.

Trazabilidad interna en la etapa de Implementación del Dominio:

- De componente reutilizable a interfaces: a partir de la implementación en el código fuente, se implementaron las interfaces correspondientes, por ejemplo: el componente *AudioMaker*, corresponde a la interfaz implementada para la clase *ConcreteAudioMakerA*.
- De componente reutilizable a generador de código: No se ha implementado aún un generador de código que satisfaga los componentes reutilizables implementados para la LPS, por lo que se debe hacer a mano el proceso de instanciación de componentes reutilizables hacia un producto en particular.
- De componente reutilizable a documentación de implementación: efectivamente se documentó la implementación asociada a los componentes reutilizables implementados por el proyecto de Ana Montero. Esta documentación fue presentada en el proyecto de título de Ana Montero.

Para el análisis de la trazabilidad vertical entre la fase de Ingeniería del Dominio e Ingeniería de la aplicación se debe considerar que en el capítulo 1.1 del proyecto de título de Ana Montero, es decir en la definición del proyecto, se define lo siguiente: “En base al estudio de la aplicación original *CollabMap*, se desarrollaron nuevos componentes de software para ser integrados en la aplicación original y que permitan complementar de mejor forma el trabajo de bomberos ante un incendio”. Esto, se traduce, para el modelo de trazabilidad, en una trazabilidad vertical la fase de Ingeniería del Dominio y la fase de

Ingeniería de la Aplicación, es decir, una instanciación de artefactos de Ingeniería del Dominio, hacia un producto en particular.

- De requisitos a requisitos particulares: Por lo que se definió anteriormente, si se cumplió la traza vertical entre los Requisitos a los Requisitos particulares, específicamente para el producto denominado *CollabMap.*
- De requisitos funcionales a requisitos funcionales particulares: se instanciaron implícitamente los requisitos funcionales hacia el producto en particular. Implícitamente quiere decir, que no se documentó específicamente como instanciación del producto en particular, pero sí ocurrió efectivamente en el producto *CollabMap*.
- De requisitos no funcionales a requisitos no funcionales particulares: se instanciaron implícitamente los requisitos no funcionales hacia el producto en particular. Implícitamente quiere decir, que no se documentó específicamente como instanciación del producto en particular, pero sí ocurrió efectivamente en el producto *CollabMap*.
- Del modelo de características a la configuración del modelo de características: efectivamente se escogieron ciertos elementos del modelo de características y se implementaron hacia un producto en particular, lo que se traduce en la instanciación del Modelo de características a la configuración del modelo de características.

Análisis de trazabilidad interna en etapa de Requisitos de la Aplicación:

- Requisitos particulares a documentación de requisitos de aplicación: Si bien, no se encuentra explícitamente como la documentación del producto en particular, es la misma que Ana define en su proyecto como especificación de requisitos, ya que será instanciada completamente hacia el producto en particular.

Análisis de la trazabilidad vertical entre la etapa de diseño de dominio y Diseño de implementación:

- De la estructura a la estructura del producto: se instancian los componentes y relaciones de la Estructura, hacia el único producto en particular, denominado *CollabMap*. Se implementó a su vez, el comportamiento del producto.

Análisis de la trazabilidad vertical entre la etapa de Implementación de dominio e Implementación de aplicación:

- De componentes reutilizables a componentes reutilizables del producto: se instanciaron los componentes reutilizables del producto hacia el producto en particular, tal como se describió en el Modelo de trazabilidad.

Para las etapas de Pruebas del dominio y Pruebas de la aplicación se concluye que; no se creó explícitamente un Plan de pruebas de dominio, ni se definió cuáles eran las pruebas de dominio que se debían ejecutar. Sin embargo, sí se realizaron pruebas de aplicación. Estas pruebas, fueron más bien pruebas de usabilidad, las cuales se hicieron para comprobar que la aplicación construida en la LPS funcionara correctamente. A continuación, se detallan en una lista de acciones, las pruebas de aplicación que se realizaron en el proyecto de Ana Montero.

Para las pruebas al componente de Audio se tiene lo siguiente [33]:

1. Abrir la aplicación *CollabMap* para acceder al servidor y a la base de datos.
2. Seleccionar una emergencia del listado.
3. Seleccionar “Asistir esta emergencia”.
4. Seleccionar del menú “Crear Recurso Digital”.
5. Seleccionar el recurso digital “Grabar audio”.
6. Grabar mensaje de audio y aceptar.
7. Seleccionar “Sí” en la ventana de diálogo ¿Desea compartir el recurso digital creado?
8. Llenar el formulario con mensaje opcional, seleccionar destinatarios y presionar Enviar.

9. Seleccionar “Sí” en la ventana de diálogo “¿Desea enviar el recurso digital?”
10. Esperar mensaje exitoso “¡Recurso digital enviado!”.

La secuencia anterior, se ejecutó en el mismo orden en que están listadas las acciones y se cumple lo siguiente para cada uno de los 5 dispositivos móviles de la tabla 1.

N° acción	Motorola	Huawei	Lenovo	Own	LG
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓
6	×	✓	✓	✓	✓
7	×	✓	✓	✓	✓
8	×	✓	✓	✓	✓
9	×	✓	✓	✓	✓
10	×	✓	✓	✓	✓

Tabla 1: Análisis sobre qué acciones fueron exitosas durante la prueba de audio

Se observa de la Tabla 1 que los dispositivos móviles Huawei, Lenovo, Own y LG, funcionan sin problemas tanto en la creación del audio como en el envío de éste. Sin embargo, el smartphone Moto G3, no cumple con iniciar el grabador, debido a que el dispositivo no tiene incorporado el grabador de audio que provee Android. Este error se produjo luego de una actualización el sistema del smartphone mencionado.

Por otro lado, si el usuario en la acción 7 de la secuencia selecciona la opción contraria, es decir “No”, se observó que la grabación se guardó satisfactoriamente en cada base de

datos local de su respectivo dispositivo móvil, esto es, en el almacenamiento predeterminado del smartphone. Así mismo, si el usuario selecciona “No” en la acción 9 de la secuencia de opciones, se cancela el envío del recurso digital, guardándose solo en la base de datos local de los dispositivos.

Para las pruebas al componente de ruta hacia la emergencia se tiene lo siguiente [33]:

A continuación, se darán a conocer la secuencia de acciones, con respecto al funcionamiento de la aplicación y de aquello que se espera que el componente de ruta hacia la emergencia haga durante la ejecución del sistema:

1. Abrir la aplicación *CollabMap* para acceder al servidor y a la base de datos.
2. Seleccionar una emergencia del listado.
3. Seleccionar del menú “Mostrar Ruta hacia la emergencia”.
4. Visualizar ruta desde la ubicación actual del usuario hasta la ubicación de la emergencia.

La secuencia anterior se ejecutó en el mismo orden en que están listadas las acciones y se cumple lo siguiente para cada uno de los 5 dispositivos móviles de la tabla 2.

N° acción	Motorola	Huawei	Lenovo	Own	LG
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓

Tabla 2: Análisis sobre qué acciones fueron exitosas durante la prueba de ruta

Las pruebas que se realizaron para este componente resultaron satisfactorias con respecto a la secuencia de acciones listada anteriormente. Mencionar, además, que, al momento de desviarse de la ruta, la aplicación se actualiza calculando una nueva ruta

respecto de la nueva ubicación del usuario. Este nuevo cálculo de ruta sucede al momento de la sincronización de la aplicación, tanto automática como manual, esto último, presionando el botón de actualización ubicado en la esquina superior derecha.

Finalmente, si el usuario visualiza otra sección o lugar en el mapa y pierde la visualización de la ruta, es posible volver a la ubicación actual del usuario seleccionando la opción “Ver mi ubicación”. Esto se pudo observar en cada uno de los dispositivos móviles en los que se ejecutó la aplicación.

7.2. Conclusión del análisis del caso de estudio

De acuerdo a la evolución que se realizó en el caso de estudio respecto al modelo de trazabilidad propuesto en este proyecto de título, se puede concluir lo siguiente:

- Se utilizó uno de los tres escenarios de evolución definidos para el modelo de trazabilidad, éste es el escenario de evolución inserción.
- Las dos características implementadas son consideradas como activos bases fundamentales dentro de la etapa de Ingeniería de dominio; esto se debe a que estos componentes reutilizables podrán ser instanciados hacia un producto en particular mediante un generador de código, el cual permitirá realizar este proceso de manera automática o semiautomática, dependiendo de cuántos desarrollos personalizados sean necesarios en un futuro. Sin embargo, para este caso de estudio, la instanciación del producto particular fue creada en forma manual, es decir, no se utilizó un generador de código para realizarla. La razón, es que éste aún no ha sido implementado como activo base dentro de la fase de Ingeniería del dominio, específicamente en la etapa de Implementación del dominio.
- La trazabilidad vertical que se define en el modelo de trazabilidad, respecto del caso de estudio, trata de la instanciación de todos los artefactos reutilizables implementados en la fase de Ingeniería del dominio hacia un producto en particular,

en la fase de Ingeniería de la aplicación. Esto se debe, a que la LPS para el caso de estudio, contempla solo un producto particular que contiene todos los artefactos del dominio.

8. Conclusiones

Se cumplió con el objetivo general propuesto al comienzo de este proyecto, "crear un modelo de trazabilidad que apoye la evolución de una línea de productos de software". Para el logro de este objetivo, se abordaron ciertos objetivos específicos; tales como: entender el proceso productivo de una línea de productos de software, entender concepto de trazabilidad en línea de productos de software, determinar posibles eventos de evolución que pueden afectar la trazabilidad en una línea de productos de software, crear modelo de trazabilidad y, por último, Validar modelo de trazabilidad.

Cabe mencionar que se presentaron ciertas dificultades para cumplir con algunos de los objetivos específicos mencionados anteriormente, principalmente para llevar a cabo la construcción del modelo de trazabilidad, debido a que se debió construir un Modelo por cada escenario de evolución definido en este proyecto (inserción, modificación y eliminación), considerando a su vez las tres dimensiones de trazabilidad (interna, horizontal y vertical).

El modelo de trazabilidad propuesto en este proyecto de título sirve como guía u orientación para la evolución de una LPS. Esto quiere decir, que también funciona para un grupo de personas que no necesariamente poseen una gran experiencia en mantención de sistemas, debido a que de acuerdo al escenario de evolución en el que se encuentren (inserción, modificación o eliminación), podrán ir siguiendo las trazas del modelo para saber con mayor facilidad dónde deberán realizar alguna otra modificación. Esto, finalmente se traduce, en una disminución, tanto en el esfuerzo, como en el tiempo que se requiere para evolucionar una Línea de productos de software.

El Modelo de trazabilidad posee las siguientes limitaciones:

- Nivel de granularidad medio: aún se pueden determinar otros artefactos dentro de los ya existentes con el objetivo de tener un Modelo de trazabilidad más

detallado; por ejemplo: las características pueden ser variables o comunes, las restricciones pueden ser de tipo excluye o incluye, entre otras.

- El Modelo de trazabilidad no se hace cargo de todos los artefactos existentes: Debido al tiempo acotado para la realización del proyecto, se consideraron para la construcción del Modelo solo los artefactos reutilizables más relevantes.

8.1. Trabajos futuros

Se puede ampliar el nivel de granularidad de los artefactos que componen el modelo de trazabilidad, con el objetivo de lograr una trazabilidad más eficiente como apoyo a la evolución de una LPS. Por ejemplo, se puede dividir el artefacto de componentes reutilizables, para lograr una trazabilidad en la codificación, lo cual permitiría modificar el código fuente con mayor facilidad, como se muestra en la figura 88.

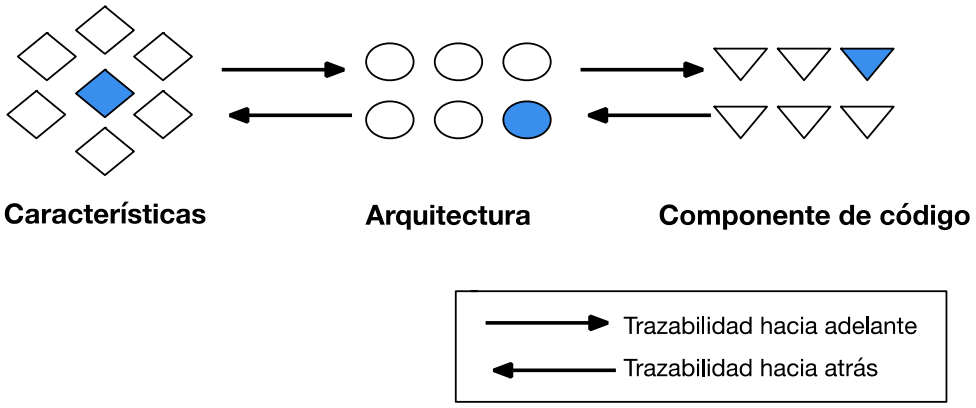

Figura 88: Ejemplo de modelo de trazabilidad para trabajos futuros

El Modelo de trazabilidad propuesto en este proyecto de título, podrá implementarse como una herramienta de trazabilidad computacional, con la cual, se podrían alcanzar los siguientes objetivos:

- Lograr realizar la trazabilidad en una línea de producto de software. Actualmente, es muy complicado lograr una trazabilidad en una LPS, incluso, a veces se hace imposible, dependiendo del tamaño de ésta.

- Disminuir significativamente los tiempos que se requieren para modificar algún artefacto dentro de la LPS. Al momento de modificar un artefacto dentro de la LPS, la herramienta de trazabilidad, podrá mostrar inmediatamente la traza que se debe cumplir, respecto del artefacto en evolución. Finalmente, se traduce, en una disminución del esfuerzo que necesita para evolucionar la LPS.

9. Glosario

Framework: es un esquema o patrón, que puede ser usado para llevar a cabo el desarrollo o implementación de una aplicación [36].

Smartphone: es el teléfono inteligente que ha sido utilizado para llevar a cabo las pruebas de la aplicación en el proyecto de Ana Montero.

Componente reutilizable: para este proyecto, corresponde a los módulos de código y las bibliotecas implementadas, con las cuales son construidas las aplicaciones en la fase de Ingeniería de la aplicación.

Modularidad: consiste en dividir un programa en módulos de código, los cuales podrán compilarse por separado. Además, los módulos de código pueden tener conexiones con otros módulos [24].

Bibliografía

[1] T. VALE, E. SANTANA DE ALMEIDA, V. ALVES, U. KULESZA, N. NIU y R. DE LIMA, «Software product lines traceability: A systematic mapping study,» *Information and Software Technology*, p. 8, 16 diciembre 2016.

[2] R. PEREIRA DE OLIVEIRA y E. SANTANA, Requirements Evolution in Software Product Lines: An Empirical Study, El salvador: conference publishing services, 2015, pp. 1-10.

[3] C. THAO, «Managing Evolution of Software Product Line,» de *34th International Conference on Software Engineering (ICSE)*, Milwaukee, 2012.

[4] O. DÍAZ y S. TRUJILLO, «Líneas de producto de Software,» Fábricas de Software: experiencias, tecnología y organización, pp. 2-7, 2010.

[5] S. PUERTA, Soporte a la trazabilidad en el desarrollo de líneas de productos software, Madrid, 2011, p. 1.

[6] P. CLEMENTS y L. NORTHROP, Software Product Line: Practice and Patterns, A. Wesley, Ed., 2001, p. 563.

[7] G. BOTTERWECK y A. PLEUS, «Evolution of Software Product Lines,» Limerick, 2014, p. 7.

[8] J. KIM, S. KANG y J. LEE, «A Comparison of Software Product Line Traceability Approaches from End-to-End Traceability Perspectives,» *World Scientic Publishing Company*, vol. 24, nº 4, 2014.

[9] F. GARCÍA, J. A. BARRAS, M. LAGUNA y J. MARQUEZ, «Líneas de producto, Componentes, Frameworks y Mecanos,» 2002.

[10] F. PINHEIRO, «Requirements Traceability,» de *Perspectives on Software Requirements*, vol. 753, J. C. Sampaio y J. Horario, Edits., Springer US, 2004, pp. 91-113.

[11] O. GOTEL y A. FINKELTEINS, «A.C.W. An analysis of the requirements traceability problem,» 1994, pp. 90-92.

[12] F. PINHEIRO y J. GOGUEN, «An object-oriented tool for tracing requirements,» de *Formal and informal aspects of requirements traceability*, 1996, pp. 52-64.

[13] R. J. WIERINGA, «An introduction to requirements traceability,» 1995.

[14] A. PLEUSS, G. BOTTERWECK, D. DHUNGANA, A. POLZER y S. KOWALESKI, «level, Model-driven support for product line evolution on feature,» *The Journal of Systems and Software*, pp. 2261-2262, 2012.

[15] J. D. STERMAN, «A Skeptic's Guide to Computer Models,» *Managing a Nation: The Microcomputer Software Catalog*, pp. 2-3, 1991.

[16] I. JACOBSON, G. BOOCH y J. RUMBAUGH, El proceso unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000.

[17] S. COLUNGA y J. GARCÍA, «monografías,» [En línea]. Available: http://www.monografias.com/trabajos36/los-modelos/los-modelos2.shtml.

[18] O. D. N. U. P. L. A. AGRICULTURA, «modelos y su uso,» Departamento de agricultura, 2016. [En línea]. Available: http://www.fao.org/docrep/w7452s/w7452s01.htm.

[19] M. SILVIA, A. BARRERA, J. ARROYAVE y J. PINERA, «Un método para la trazabilidad de requisitos en el proceso unificado de desarrollo,» pp. 69-82, diciembre 2007.

[20] K. POHL, G. BÖCKLE y F. VAN DER LINDEN, Software Product Line Engineering, S. B. Heidelberg, Ed., 2005, p. 124.

[21] P. O. ROSSEL, «Software Product Line model for the meshing tool domain,» Santiago, 2013.

[22] K. POHL y T. WEYER, «Documenting Variability in Requirements Artefacts,» S. B. Heidelberg, Ed., 2005, p. 92.

[23] V. LAMSWEERDE, «The Requirements Engineering Framework,» S. B. Heidelberg, Ed., 2001, p. 53.

[24] F. VAN DER LINDEN, K. SCHMID y E. ROMMES, Software Product Lines in Action, S. B. Heidelberg, Ed., 2007, p. 38.

[25] E. A. JAZAYERI, Software Product Line Engineering, Springer ed., 2000.

[26] A. VAN DEURSEN, P. KLINT y J. VISSER, «Domain-Specific Languages: An Annotated Bibliography,» pp. 26-36, 2000.

[27] K. POHL y E. SIKORA, Documenting Variability in Test Artefacts, S. B. Heidelberg, Ed., 2005, pp. 151-152.

[28] F. VAN DER LINDEN, Software Product Families in Europe: The Esaps & Café Projects, 2002, pp. 41-49.

[29] F. VAN DER LINDEN, Domain Design, S. B. Heidelberg, Ed., 2005, pp. 218-240.

[30] K. POHL y A. REUYS, «Application testing,» de *Software Product Line Engineering*, 2005, p. 357.

[31] I. JACOBSON, P. JOHNSON, M. CHRISTERSON y G. OVEERGARD, Ingeniería de software orientada a objetos, un acercamiento a través de los casos de uso, A. Weasley, Ed., Upper Saddle River, New Jersey, 1992.

[32] S. BÜHNE y K. POHL, «Domain Requirements Engineering,» de *Software Product Line Engineering*, S. B. Heidelberg, Ed., 2005, pp. 194-216.

[33] A. MONTERO CÁCERES, «Mantención de una familia de productos en el dominio de administración de emergencias,» Concepción, 2017.

[34] E. ORMEÑO, «Construcción de componentes reutilizables para la elaboración de aplicaciones móviles que apoyen el trabajo colaborativo en emergencias,» Concepción, 2015.

[35] P. O. ROSSEL, V. HERSKOVIC y E. ORMEÑO, «Creating a family of collaborative applications for emergency management in the firefighting sub-domain,» *Springer Science+Business Media New York 2015,* 2015.

[36] K. MOHAN y B. RAMESH, «Managing Variability with Traceability in Product and Service Families,» *Proceedings of the 35th Hawaii International Conference on System Sciences - 2002,* 2002.

[37] J. LEE y S. HWANG, «Variability Change Management Using the Orthogonal

Variability Model-Based Traceability,» *CrossMark*, 29 enero 2016.

[38] K. CZARNECKI y U. W. EISENECKER, Generative Programming. Methods, Tools, and Applications, Addison Wesley, 2000.

[39] L. NORTHROP y P. CLEMENTS, «A Framework for Software Product Line Practice,» 2007. [En línea]. Available: http://www.sei.cmu.edu/productlines/framework.html.

[40] Y. GHANAM y F. MAURER, Linking Feature Models to Code Artifacts Using Executable Acceptance Tests, Alberta, 2010, p. 222.

[41] S. KANG, «Requirements for systematic software product line development,» 2011.

[42] E. LÓPEZ L, M. GONZÁLEZ, M. LÓPEZ S y E. IDUÑATE, «Proceso de desarrollo de software mediante herramientas MDA,» vol. 3, Cuernavaca, 2006.

[43] A. MACCARI, Experiences in assessing product family software architectures for evolution, 2002, p. 585–592.

[44] J. MORECROFT, System Dynamics and Microworlds for Policy Makers, 1988, pp. 301-302.

[45] D. WEISS y R. LAI, «Software Product-Line engineering: A Family Based Software Development Process,» 1999.

[46] K. MOHAN, Managing Variability with Traceability in Product and Service Families, c. society, Ed., Georgia, 2002, p. 2.

[47] P. GARCÍA, Aplicación de ingeniería dirigida por modelos (MDA), para la construcción de una herramienta de modelado de dominio específico (DSM) y la creación de módulos en sistemas de gestión de aprendizaje (IMS) independientes de la plataforma, vol. 78, DYNA, 2011.

[48] J. A. HURTADO, «a meta-process for defining adaptable software processes,» santiago, 2012.