# Análisis Visual de la Modularidad de Modelos de Procesos de Software
# AVIMO-PS


Monografía de Trabajo de Grado para Optar al Título de
Ingeniero de Sistemas

**Fredy Alberto Cárdenas Bolaños**
**Jhonattan Solarte Martínez**

**Director: PhD. Julio Ariel Hurtado**

*Universidad del Cauca*

**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Departamento de Sistemas**
**Grupo IDIS – Investigación y Desarrollo en Ingeniería de Software**
**Línea de Investigación en Procesos Software**
**Popayán, Mayo de 2013**

# TABLA DE CONTENIDO

# LISTA DE FIGURAS

# LISTA DE TABLAS

# 1. INTRODUCCION

## 1.1 Contexto del problema

Existe un acuerdo generalizado en la industria de software sobre la importancia de contar con modelos de proceso de software bien definidos al interior de las organizaciones desarrolladoras de software [1], con el fin de llegar y mantenerse en un mercado competitivo tanto a nivel regional como internacional. Al igual que en la mayoría de los sectores industriales, se reconoce que la forma en que se construye el software (procesos de software) afectan la productividad y la calidad del producto [2]; es decir, al tener los procesos controlados se puede obtener una mejor gestión de las cualidades requeridas de los productos de software. Sin embargo, esto fue informalmente reconocido hasta finales de los noventa [3]. A partir de entonces, los modelos de proceso de software han sido reconocidos formalmente por investigadores como un de tema de vital importancia, y desde entonces han buscado comprender su aplicación y utilidad, así como identificar los métodos y su calidad [4]. Desde la perspectiva industrial, las organizaciones se vienen esforzando por conseguir procesos definidos. Normas como ISO/IEC15504[1] y modelos de madurez como CMMI[2], son generalmente utilizados como guías para mejorar los procesos de las organizaciones de software.

Actualmente, las empresas desarrolladoras de software están mejorando o adoptando procesos adecuados a sus necesidades, con el fin de reducir los costos, mejorar la productividad, la gestión y el manejo del tiempo [5]. Esto ha dado lugar a que cada vez se emplee una mayor cantidad de tiempo a la definición, mejora y medición del proceso de software, en particular para evaluar y predecir la calidad del mismo [6]. Debido al esfuerzo dedicado al proceso, los modelos cobran una vital importancia. Y es a través de ellos que el proceso se describe, y es presentado a los diferentes participantes involucrados (ingenieros de software). Dado el costo de su definición y del impacto que tienen sobre la producción de software, es necesario evaluar aspectos del proceso que permitan conocer su calidad. Por ejemplo, si está correctamente modelado, si es fácil de modificar y si es fácil de entender por parte de los usuarios, entre otros aspectos [7].

Para la evaluación de los modelos de procesos existen varias alternativas, que incluyen la prueba o ensayo y error (proponer, mejorar y probarlas en proyectos piloto)[8] [9], la

[1] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54537
[2] http://www.sei.cmu.edu/cmmi/index.cfm

simulación (su ejecución en máquinas intérpretes de procesos) [10], las métricas [11], las verificaciones formales [12] y la visualización [13]. La visualización de modelos de proceso, se ha mostrado como una alternativa viable y efectiva para el análisis de modelos de procesos, como es el caso de los Blueprints [1] y los patrones de errores propuestos por AVISPA (Analysis and Visualization for Software Process Assessment) [13].

Los blueprints son vistas de modelos de proceso de software que facilitan intuitivamente, a los diseñadores de procesos en primera instancia identificar las anomalías o problemas en la especificación y/o el modelo, proporcionando pistas de cómo encontrar las inconsistencias en un modelo de proceso de software y a partir de éstas, mejorar la calidad de los modelos de procesos de software. Los patrones de errores de AVISPA permiten identificar de manera concreta problemas específicos en el modelo de proceso, resaltándolos y diferenciándolos de los demás [14].

### 1.1.1 Problema

Los modelos de procesos de software son representaciones explícitas del conocimiento organizacional, que buscan guiar y administrar sus proyectos de software. Dichas representaciones evolucionan debido a cambios en los negocios, las estructuras organizacionales, los equipos y la naturaleza variable de los proyectos. Dado que los modelos de procesos de software se definen como una base para reutilizar conocimiento y experiencia, es importante conocer si los “modelos de proceso” están bien estructurados para soportar el cambio y evolución. Los procesos de software juegan un papel importante para las empresas desarrolladoras de software que adoptan iniciativas de mejoramiento, dado que estos procesos deben modificarse y evolucionar sistemáticamente, guiándose por nuevos objetivos organizacionales para adaptarse y satisfacer los nuevos escenarios y su entorno tan cambiante [15] [16]. Sin embargo, los modelos de proceso normalmente son diseñados para cubrir aspectos de uso y no para cubrir aspectos de evolución como la reutilización, la modificabilidad y la adaptabilidad. Un modelo de proceso difícil de mantener, es un problema significativo para las organizaciones, porque su evolución requiere de un mayor esfuerzo y además es propenso al error.

Kaur [17] afirma que la evolución del software en sí, es un proceso que puede ser prototipado, desarrollado sistemáticamente, (re)configurado, medido, refinado, administrado y mantenido, representando una secuencia en red de actividades, objetos, transformaciones y eventos que incorporan estrategias para el logro de la evolución del software. Es aquí, donde la modificabilidad tiene un papel muy importante, siendo un atributo de calidad que concierne a la capacidad de un producto para ser cambiado con facilidad [18]. De forma similar al software, la modificabilidad de un proceso, es la capacidad que tiene para adaptarse en el tiempo (cambio del proceso en su evolución a través de ciclos de mejora) y en el espacio de procesos (adaptaciones a contextos específicos) [18]. Evaluar qué tan modificable es un modelo de procesos es una necesidad relevante para el equipo de ingeniería de procesos. Sin embargo, los esfuerzos

de evaluación han prestado poco énfasis en otros aspectos más allá de la correctitud y usabilidad de los modelos de proceso. Este también es el caso de los Blueprints y patrones de error propuestos en AVISPA, los cuales están focalizados a evaluar lo correcto del modelo de proceso, en términos de elementos individuales y relacionados de los procesos, es decir, respecto de sus tareas, roles y productos de trabajo, dejando por fuera otros aspectos relevantes para la calidad de los modelos, como es el caso de la modificabilidad. Para el caso del software se ha demostrado una estrecha relación entre la modularidad y modificabilidad [19] [83]. Dado que el proceso de software es un dominio específico de software [20], evaluar la modularidad de un modelo de proceso de software, permite evaluar si el modelo de proceso facilita la modificación y su evolución. Por tanto, las métricas definidas para evaluar la modularidad del software pueden ser adaptadas para definir la modularidad de los modelos de proceso y a partir de ellas obtener nuevos Blueprints y patrones de error.

### 1.1.2 Pregunta de Investigación

Por lo anterior y teniendo en cuenta, que los procesos de software juegan un papel importante para las empresas desarrolladoras de software y que es elemental conocer la capacidad de un modelo de proceso para que pueda ser mantenido fácilmente, surge la siguiente pregunta de investigación:

¿Cómo identificar visualmente si un modelo de proceso de software ha sido modularmente bien definido para facilitar su cambio y evolución?

## 1.2 Objetivos

### 1.2.1 Objetivo General

- Analizar visualmente la modularidad de los modelos de proceso SPEM 2.0 construidos con Eclipse Process Framework - EPF.

### 1.2.2 Objetivos Específicos

- Definir y diseñar un modelo de métricas para analizar la modularidad de los modelos de proceso.

- Desarrollar un prototipo denominado “AVIMO-PS” (Análisis Visual de la Modularidad de Modelos de Procesos de Software), herramienta que extiende a AVISPA, en la cual se implementan métricas de modularidad definidas.

- Implementar en el prototipo software, un conjunto de Blueprints que faciliten la visualización de la modularidad del modelo de proceso basado en el modelo de métricas definido.

- Evaluar la propuesta usando dos modelos de proceso industriales y dos de código abierto e identificar al menos un patrón problemático asociado a la modularidad.

## 1.3 Propuesta y Justificación

Las empresas de desarrollo de software que desean implementar proyectos de SPI (Software Process Improvement - Mejora de Procesos de Software), están en un contexto muy cambiante y al igual que sus proyectos de desarrollo, necesitan reducir drásticamente los tiempos y costos relacionados con la producción, sin alterar la calidad del producto[4]. La mejora de procesos organizacionales está ganando peso porque los procesos de desarrollo y mantenimiento de software se han convertido, cada vez más, en actividades complejas y laboriosas de carácter intelectual, con un alto potencial para incrementar la competitividad organizacional [5]. Sin embargo, garantizar la calidad de los procesos en el marco de estas mejoras no es una tarea fácil, para ello se propone el análisis de modelos de procesos de software como mecanismo para asegurar la calidad en las especificaciones de los procesos. El análisis de procesos basado en la visualización permite a los ingenieros de procesos detectar problemas y analizar datos en modelos definidos de manera temprana, utilizando la capacidad humana para visualizar e interpretar las vistas de los modelos de procesos, logrando evaluar la calidad de un proceso de software definido estáticamente en una forma temprana, justo antes de ponerlos a prueba en proyectos reales, y con ello tomar decisiones para mitigar riesgos por los posibles errores(debido a la mala especificación) en su aplicación.

Expertos en ingeniería están de acuerdo en la conveniencia de especificar las arquitecturas de software con múltiples vistas [21]. El presente proyecto sigue el mismo enfoque pero orientado a modelos de procesos de software y particularmente a la modularidad, dado a que éste es un concepto fundamental de la ingeniería del software y para lograr buena modificabilidad, esto comprende métricas tales como son el acoplamiento, la cohesión y la inestabilidad [22]. Debido a lo anterior, es importante evaluar la facilidad de modificación de un modelo de proceso software a través de una medición y representación visual del nivel de modularidad con que éste fue definido. Expertos en ingeniería de software aseguran que los diseños con bajo acoplamiento y alta cohesión dan lugar a productos que son a la vez, más fiables y más fáciles de mantener [6] [23]; la modularidad de un modelo de proceso software puede ser obtenida a partir de métricas más específicas como el acoplamiento y la cohesión [24]. Como se mencionó anteriormente la visualización de procesos de software se utiliza poco en nuestra industria, por lo tanto, los resultados de este trabajo aportarán un grano de arena para

optimizar el diseño y elaboración de procesos de software con el fin de reducir los costos, mejorar la productividad, la gestión y el manejo del tiempo [25].

Es importante ayudar y asistir a los ingenieros de procesos a evaluar la calidad de los modelos de procesos de software, determinando el impacto que conlleva realizar cambios en los modelos de procesos de software definidos, reconociendo de manera temprana los errores presentados en los modelos de procesos modificados, por lo anterior se construirá la herramienta AVIMO-PS con el fin de facilitar la tarea de revisión, aplicando métricas que permitan realizar el análisis visual de la modularidad, tales como métricas de acoplamiento y cohesión de paquetes de métodos en modelos de proceso SPEM 2.0[3]. Este proyecto busca integrar las buenas prácticas citadas, ampliamente usadas y difundidas en el área de la mejora de procesos de software, y de esta manera enriquecer el desarrollo de nuevo conocimiento para la comunidad académica y científica internacional. Este conocimiento es de tipo exploratorio y descriptivo.

Este proyecto de grado busca hacer un aporte significativo a la academia, especialmente en la línea de investigación de Ingeniería del Software del grupo IDIS (Investigación y Desarrollo en Ingeniería del Software) [26], puesto que su ejecución requiere de la utilización de conocimientos informáticos, así como también el conocimienton de temas relacionados con la Ingeniería del Software enfatizando en temas específicos como métodos de evaluación visual de procesos de software.

## 1.4 Organización del Documento

Este documento se encuentra estructurado de la siguiente manera: el capítulo 2 incluye el estado del arte y los trabajos relacionados sobre la evaluación de los modelos de procesos software, el estándar SPEM2.0, y los procesos de software. El capítulo 3 presenta las métricas y los Blueprints propuestos para el análisis de la modularidad de los modelos de procesos de software, y su implementación en el prototipo AVIMO-PS que es una extensión de AVISPA para la evaluación visual de modelos de procesos, teniendo en cuenta la modularidad de los paquetes de contenido de método. El capítulo 4 se presenta el estudio de caso donde se evalúan dos modelos de proceso industriales y dos de código abierto usando el prototipo AVIMO-PS. Finalmente el capítulo 5 presenta las conclusiones, limitaciones y trabajo futuro planteado sobre este trabajo de grado.

[3] http://www.omg.org/spec/SPEM/2.0/

# 2 MARCO TEÓRICO

Una de las principales líneas de trabajo para la mejora de la calidad de los productos software es el estudio y la mejora de los procesos, mediante los cuales el software es desarrollado y mantenido. Esto es debido a que existe una correlación directa entre la calidad del producto y la calidad del proceso utilizado para su fabricación [5]. Es decir ahora es importante entender y evaluar la calidad del proceso software a través de sus representaciones (modelo de proceso).

## 2.1 EL PROCESO DE SOFTWARE

La ingeniería del software adquirió importancia como una disciplina cuando el desarrollo de software se reconoció como una tarea muy compleja, hasta tal punto que ya no fue confiable hacer producir software de forma artesanal y de forma dependiente exclusivamente de las personas. Durante las últimas cuatro décadas diferentes métodos y técnicas han sido desarrollados con el ánimo de facilitar el desarrollo de sistemas. El concepto clave en éste trabajo de grado es el proceso de software, el cual es definido por Fugguetta [27], como: “Un conjunto coherente de reglas, políticas, actividades y procedimientos, componentes de software, metodologías y herramientas usadas o creadas específicamente para concebir, desarrollar, instalar y mantener un producto software”.

El desarrollo de software relaciona un conjunto de acciones y actividades que permiten transformar los requerimientos de un usuario en una solución software, en donde intervienen áreas importantes como son el análisis de la información, el desarrollo técnico, el control de calidad, la gestión de proyectos, las actividades de atención al cliente, entre otras. Una característica usual a los procesos de software es la particularidad de que en el mercado actual de la última década ha conducido a los ingenieros, desarrolladores, analistas de negocio y a las organizaciones en general, a centrarse en sus procesos como referencia para progresar y subsistir [28]. Esta situación ha incrementado la necesidad de analizar, evaluar, medir y mejorar los procesos de software. Durante la definición de un proceso se software se identifican roles, tareas y artefactos que son el conjunto de elementos claves de un modelo de proceso[29].

### 2.1.1 Características de los procesos de software

Hay diversas características importantes al trabajar con procesos de software, a continuación se mencionan algunas de ellas [30]:

- La complejidad es una característica que los procesos exhiben de forma similar al software.
- No son procesos de ingeniería 'pura', debido a que no aplican abstracciones estándares (no hay fundamentos empíricos/teóricos universales sobre los cuales apoyarse), dependen en gran medida de los diseñadores.
- El diseño y producción de los procesos de software no están alineados con los presupuestos, cronogramas y calidad requerida; por lo anterior no pueden ser planificados de forma suficientemente confiable.
- Los procesos están dirigidos por excepciones, los cuales están determinados a entornos impredecibles y cada uno tiene una característica que lo diferencia de los demás.

De acuerdo a Acuña*, et al.* [25], el proceso de software definido busca garantizar:

- Una comprensión efectiva del proceso a seguir durante el desarrollo de software con los usuarios, clientes, desarrolladores, gerentes, entre otros.
- Una reutilización del conocimiento a través de los procesos, minimizando los costos relacionados a la redefinición de procesos.
- Una comprensión más formal a la gerencia, con el fin de facilitar la automatización del proceso permitiendo la independencia de los individuos.
- Permite la evolución del proceso, facilitando los medios necesarios para el aprendizaje del proceso y un sólido fundamento para su mejoramiento.

Para el desarrollo de productos de software se deben tener en cuenta [31]:

- **Personal**: con conocimiento, experiencia y competencia para el desarrollo y evolución del producto. Sin este elemento, es complicado generar productos que satisfagan las expectativas de los clientes.
- **Tecnologías**: para el desarrollo y puesta en producción de los sistemas.
- **Procesos**: que integran la experiencia del personal, los métodos y las tecnologías de forma efectiva y eficiente para elaborar productos con la calidad que los clientes requieren dentro de las restricciones de costo y tiempo.

Teniendo en cuenta que el personal y la tecnología implican costos, éstos deben ser usados de forma efectiva y productiva, por lo anterior los procesos de desarrollo de software son necesarios para las empresas. Un proceso de software mal definido puede generar graves consecuencias y derivar en costos muy elevados para la organización, lo que en un mercado competitivo, determina el éxito o el fracaso [31].

### 2.1.2 Calidad de producto y calidad de proceso

Para tener una mejor idea de la calidad de proceso, es necesario inicialmente analizar que es la calidad de un producto, en nuestro caso de productos de software.

#### 2.1.2.1 Calidad de Producto

Para definir la calidad de un producto de software, inicialmente es necesario conocer el concepto de software, el cual es un conjunto de programas, documentos, procedimientos y rutinas, destinadas a ser utilizadas en un sistema informático que realizan una función o tarea para obtener un resultado determinado. De acuerdo a la norma ISO/IEC 9126 [32], la calidad de un producto de software está relacionada con que éste satisfaga las necesidades y expectativas razonables del cliente, tomando en cuenta una combinación lógica de los siguientes aspectos:

- Portabilidad.
- Funcionalidad.
- Mantenibilidad.
- Modificabilidad
- Confiabilidad.
- Eficiencia.
- Usabilidad.
- Uso de tiempo y recursos.
- Fácil de aprender, entender y usar.

#### 2.1.2.2 Calidad de Proceso

Los procesos de desarrollo de software abarcan actividades tanto técnicas como administrativas, las cuales son requeridas para la fabricación de un sistema de software. Estas actividades comprenden desde el análisis de requisitos hasta el mantenimiento del software, pasando por el diseño, pruebas, aseguramiento de calidad, implantación, entre otras. Para tener un proceso de calidad se requiere que en primera medida, esté bien definido y elaborado, y por otra parte que sirva para lo que se especificó, en otros términos, que se pueda verificar que se cumplen y satisfacen los objetivos sobre los cuales fue definido [33]. Para definir un proceso de software se debe tener en cuenta:

- **Objetivos bien definidos**: es decir, tener claro que producto se desea obtener
- **Actividades**: cuales son las secuencias de pasos que se deben seguir.
- **Métodos**: cómo se deben ejecutar las actividades.
- **Personas involucradas**: definir los roles para ejecutar actividades.
- **Tiempos de ejecución**: establecer cuando se deben ejecutar actividades.
- **Herramientas a utilizar**: herramientas o tecnologías en las cuales apoyarse para definir y ejecutar el proceso.

- **Mediciones:** la manera en que se deben medir los elementos dentro del proceso de tal forma que se puedan verificar los resultados.
- **Acciones correctivas**: los planes que se deben ejecutar cuando se enfrenta a un problema

El concepto de calidad de procesos de software ha crecido en la industria, dado que las empresas se han visto en la necesidad de contar con modelos que les permita evaluar la capacidad para generar software con calidad y de ésta forma participar en mercados que requieren productos con un alto nivel de calidad [33].

## 2.2 MODELOS DE PROCESO DE SOFTWARE

Los modelos y estándares de procesos de software establecen un conjunto de criterios que sirven para dirigir la forma de aplicar correctamente la ingeniería de software, de no seguir alguna metodología muy probablemente habrá un nivel de calidad muy bajo en los productos. Actualmente las metodologías existentes no imponen las actividades específicas para ejecutar de manera concreta un modelo, por ello cada empresa debe decidir cómo utilizar los métodos, técnicas y herramientas que consideren más adecuadas a su necesidad,

Finkelstein y Kramer [34], definen un modelo de procesos como la descripción de un proceso expresado en un lenguaje de modelado de procesos adecuado. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real [35]. Los modelos genéricos no son representaciones finales de procesos de software, aún así, pueden ser usados para exponer distintos enfoques del desarrollo de software. Cada modelo establece una sucesión de fases y un encadenamiento entre ellas. De acuerdo a las fases y el modo en que se realice este encadenamiento, se tienen diferentes modelos de proceso. Un modelo es más adecuado que otro para desarrollar un proyecto especifico dependiendo de un conjunto de características de éste.

Los modelos de proceso pueden definirse desde diferentes puntos de vista, por ejemplo, un modelo puede definir los agentes involucrados en una actividad, mientras que otros pueden centrarse en definir las relaciones entre las actividades [36] [37]. Un modelo de proceso de software es una representación abstracta de la arquitectura, diseño o definición de un proceso de software [38]. Cada representación describe: diferentes niveles de detalle, una organización de elementos de un proceso finalizado o llevado a cabo y también proporciona una definición de los procesos los cuales pueden ser usados para la evaluación y la mejora. Un modelo de proceso puede ser analizado, validado y simulado si es ejecutado [39] [40].

### 2.2.1 Elementos básicos de un modelo de proceso.

Hay varios tipos de elementos que puede ser modelados, entre los cuales se encuentran las actividades, productos, recursos, roles, entre otros [40]. Existen varias clasificaciones sobre los principales elementos de un modelo de proceso [36]. Los elementos más comúnmente modelados son presentados en la Figura 1, la cual presenta los elementos y sus relaciones.

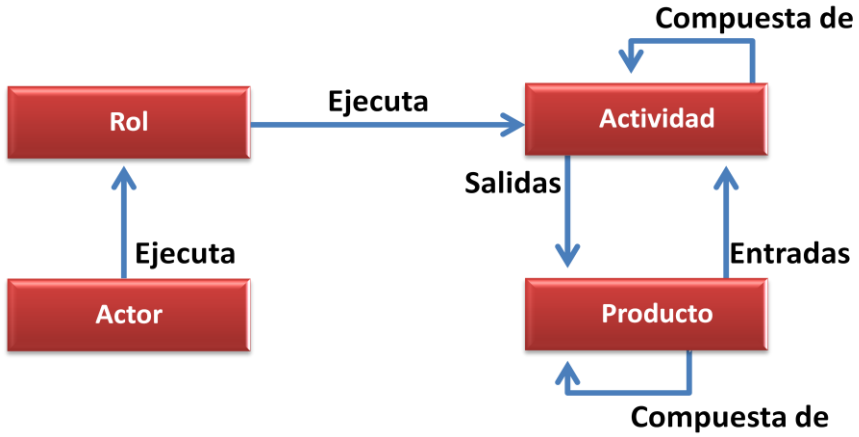



**Figura 1. Componentes básicos de modelado de procesos.**

- **Actor**: es una entidad que ejecuta un proceso. Los actores pueden ser divididos en dos grupos en cuanto a su estrecha relación con el equipo:

  a) Actores humanos, los cuales puedes ser personas que desarrolla software o están involucrados en el procesos de software y posiblemente están organizados en equipos.
  b) Los actores del sistema o herramientas del sistema, que son los programas o componentes de hardware.

  Un actor es caracterizado por las propiedades de sus funciones y sus capacidades, un actor, persona o sistema pueden desempeñar diferentes funciones, las cuales son compuestas de un conjunto consistente de actividades.

- **Rol:** Describe un conjunto de actores o grupo de responsabilidades, derechos y habilidades requeridas para ejecutar una actividad específica del proceso de software. Es una asociación entre los actores y las actividades en términos de un conjunto definido se responsabilidades las cuales son ejecutadas por actores.

- **Actividad:** Es un fragmento de un proceso que produce cambios visibles externos del estado del producto. Una actividad puede tener unas entradas, unas salidas y algunos resultados intermedios, generalmente denominados productos (los cuales pueden ser un producto software, documentos de especificación de requerimientos, esquemas de bases de datos, etc.). La actividad incluye e

implementa procedimientos, reglas, políticas y objeticos para generar y modificar un conjunto de artefactos generados. Estas actividades pueden ser ejecutadas por un actor utilizando una herramienta.

- **Producto:** Es el subproducto y la "materia prima" de un proceso. Un artefacto generado por un proceso puede ser usado más adelante como materia prima del mismo proceso o uno similar. Un artefacto o producto de software es desarrollado y mantenido en un proceso.

## 2.3 MODELOS DE REFERENCIA DE PROCESOS

Los modelos de referencia de procesos software son aquellos que definen cuáles son las mejores prácticas que una organización debe implementar para el desarrollo de software y las características que deben cumplir para obtener un determinado nivel de madurez. Estos modelos se presentan de forma genérica, lo que les permite ser adoptados e interpretados según la realidad y necesidad de las organizaciones. Entre los principales modelos se pueden mencionar: CMMI, ISO/IEC 12207[4], ISO 9001:2008[5], entre otros los cuales causan confusión si son interpretados incorrectamente.

### 2.3.1 CMMI (Capability Maturity Model Integrated) – Modelo de Capacidad y Madurez Integrado

A medida que evolucionan los procesos de una organización estos van alcanzando un nivel de madurez que les permite abordar y afrontar los proyectos de desarrollo de software. El Software Engineering Institute[6], ha generado un marco de referencia que permite evaluar el nivel de madurez de los procesos de una organización. El SEI emplea un modelo de evaluación y un esquema de cinco niveles. Este esquema establece la aprobación con un modelo de madurez de la capacidad (CMM, Capability Maturity Model), el cual define las actividades que son necesarias en los diferentes niveles de madurez del proceso. La estimación define cuales de las actividades del modelo se están ejecutando, verificando de esta manera el nivel de madurez de las organizaciones. En la actualidad, se cuenta con un modelo integrado conocido como CMMI, en el cual se han incluido diversos aspectos relacionados con el desarrollo de software. En CMMI, se establecen cinco niveles de madurez:

- Nivel 1: Inicial.
- Nivel 2: Administrado.
- Nivel 3: Definido.
- Nivel 4: Administrado cuantitativamente.

[4] http://www.12207.com/
[5] http://www.iso.org/iso/catalogue_detail?csnumber=46486
[6] http://www.sei.cmu.edu/|

- Nivel 5: Optimizado.

Para cada uno de los niveles de madurez CMMI establece áreas de proceso, las cuales están divididas en cuatro categorías: Soporte, Ingeniería, Administración de proyectos Y Administración de procesos; éstas áreas describen las actividades de ingeniería de software necesarias para cumplir una práctica en un nivel en particular. CMMI, permite determinar el nivel de capacidad y madurez de los procesos en el desarrollo de software, abarcando el ciclo de vida del producto desde su concepción, entrega y mantenimiento.

### 2.3.2 ISO/IEC 12207

La norma ISO/IEC 12207 [41], es utilizada como modelo de referencia para el ciclo de vida de software. Para la aplicación de la norma es necesario que las organizaciones definan procesos principales, procesos de apoyo y procesos organizativos. Estos procesos se dividen en actividades, y éstas a su vez se separan en tareas las cuales son ejecutadas por roles [42]. ISO/IEC 12207, no aplica limitaciones acerca de la metodología a utilizar para llevar a cabo los procesos. Por lo anterior, es viable implementar los procesos, por ejemplo con metodologías ágiles. Los procesos descritos en ISO/IEC 12207, están definidos para alcanzar los propósitos y resultados estudiando y analizando la implementación y adecuación de los procesos a una organización concreta [42]. El modelo de referencia es utilizado también para brindar una base común para diferentes modelos y métodos para asegurar que la evaluación sea llevada a cabo en un contexto común [42].

### 2.3.3 ISO 9001:2008

La ISO 9001:2008 define los requisitos necesarios para implementar un sistema de gestión de calidad, centrándose en los ítems requeridos en una organización para administrar y mejorar la calidad de sus productos y servicios, satisfaciendo al cliente cumpliendo los requisitos legales y reglamentarios aplicando eficaz y eficientemente los recursos garantizando la mejora continua. Todos los requisitos de la Norma ISO 9001:2008 son genéricos y son aplicables a todas las organizaciones, sin importar su tipo, tamaño y producto suministrado. Para que una organización funcione correctamente, debe implementar un conjunto de actividades relacionadas entre sí. Un proceso se puede considerar como un conjunto de elementos y actividades que se relacionan entre sí para transformar entradas en salidas optimizando el uso de recursos. Normalmente la salida de un procesos se transforma en la entrada del siguiente proceso [43].

## 2.4 MODELOS DE EVALUACIÓN DE LA CAPACIDAD DE LOS PROCESOS DE SOFTWARE

Los métodos de evaluación permiten conocer, por medio de la ejecución de la evaluación, la situación actual de la organización, la capacidad de los procesos y con base en estos implementar un modelo de mejora continua. Entre los principales métodos de evaluación se encuentran: ISO/IEC 15504, SCAMPI[7], EvalProSoft, entre otros.

### 2.4.1 ISO/IEC 15504

La norma ISO/IEC 15504 [44], conocida como proyecto SPICE (Software Process Improvement and Capability dEtermination) es un estándar para procesos de desarrollo software que provee de un marco de trabajo uniforme para la evaluación del proceso, y establece los requisitos mínimos para realizar una evaluación que asegure la repetitividad y consistencia de las valoraciones obtenidas. ISO/IEC 15504 es un estándar internacional de evaluación y determinación de la capacidad de los procesos involucrados en el ciclo de vida del software y mejora continua de procesos de ingeniería del software, con la finalidad de desarrollar un grupo de medidas de capacidad construidas para todos los procesos de la organización. La norma ISO/IEC 15504 define dos niveles de evaluación para mejorar los procesos[44]:

- **Niveles de capacidad:** Están formados por seis niveles de capacidad, que representan el aumento de capacidad del proceso, del 0 al 5. Los niveles de capacidad pueden establecer un camino para la mejora de cada proceso.
- **Nivel de madurez:** Donde la organización mejora los procesos obteniendo una ponderación cuyo alcance es la organización (proyectos, áreas, etc.).

### 2.4.2 SCAMPI

El método estándar de CMMI para la mejora de procesos SCAMPI (Standard CMMI Appraisal Method for Process Improvement), permite evaluar la forma de cómo una organización que trabaja con procesos y metodologías cumplen con un modelo de CMMI. La evaluación del SCAMPI, determina el nivel de capacidad o madurez que ha logrado una organización que aplica CMMI en sus procesos. El objetivo principal de la evaluación es detectar las fortalezas y oportunidades de mejora de los procesos adoptados en la organización, asociados a las prácticas específicas y genéricas que menciona el modelo de referencia. La finalidad de una evaluación con el método SCAMPI es reconocer el nivel de institucionalización de las prácticas específicas y genéricas de las áreas de proceso evaluadas y que las prácticas no solo estén descritas sino que sean aplicadas a los proyectos que se llevan a cabo en una organización [45] .

[7] http://www.sei.cmu.edu/library/abstracts/reports/01hb001.cfm

### 2.4.3 EVALPROSOFT

EvalProSoft [46], tiene como propósito la evaluación de procesos para las organizaciones de software que requieran tener un perfil de capacidad de los procesos implantados y un nivel de madurez de la organización. Esta evaluación aplica principalmente para las organizaciones que han utilizado como modelo de referencia a Moprosoft[8] para la implantación de proceso [47]. Según Oktaba [47], EvalProSoft está basado en la norma Internacional ISO/IEC 15504.

El método dispone de dos dimensiones donde se evalúa a los procesos frente a las capacidades que éstos presenten, gran parte del método se alinea a la norma ISO/IEC15504. EvalProSoft, puede ser usado de tres formas, como evaluación para acreditación de capacidades, evaluación de capacidades del proveedor y auto-evaluación de capacidades de proceso [47].

## 2.5 SPEM 2.0

Para la especificación de procesos se software, se cuenta con la segunda versión del Meta-Modelo de Ingeniería de Procesos de Software y Sistemas (Software and Systems Process Engineering Meta-Model – SPEM 2.0). SPEM 2.0 es un estándar de la OMG[9] que ofrece un marco para la definición de procesos de desarrollo de sistemas y de software, y también provee los conceptos necesarios para la definición, descripción, modelamiento y presentación de todos los elementos que los componen [48]. Con el uso de SPEM 2.0, se pueden generar modelos de procesos de software en formato procesable por la máquina, lo cual permite:

- Facilitar la interpretación e intercambio de información.
- Llevar un adecuado uso del contenido de los procesos.
- Elaborar, administrar y gestionar el crecimiento sistemático de procesos.
- Facilitar la reutilización.
- Dar soporte a la mejora de procesos.
- Guiar la automatización de procesos.

La idea central en SPEM 2.0 es representar procesos fundamentado en tres elementos básicos: rol, producto de trabajo y tarea (ver Figura 2).

- Las tareas representan el esfuerzo a realizar.
- Los roles representan quien lo realiza.

[8] http://www.moprosoft.com.mx/
[9] http://www.omg.org/

- Los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen.

La idea central de un modelo de proceso consiste, básicamente, en decir quién (rol) realiza qué (tarea) con el fin de, a partir de unas entradas (productos de trabajo) obtener unas salidas (productos de trabajo) [48].

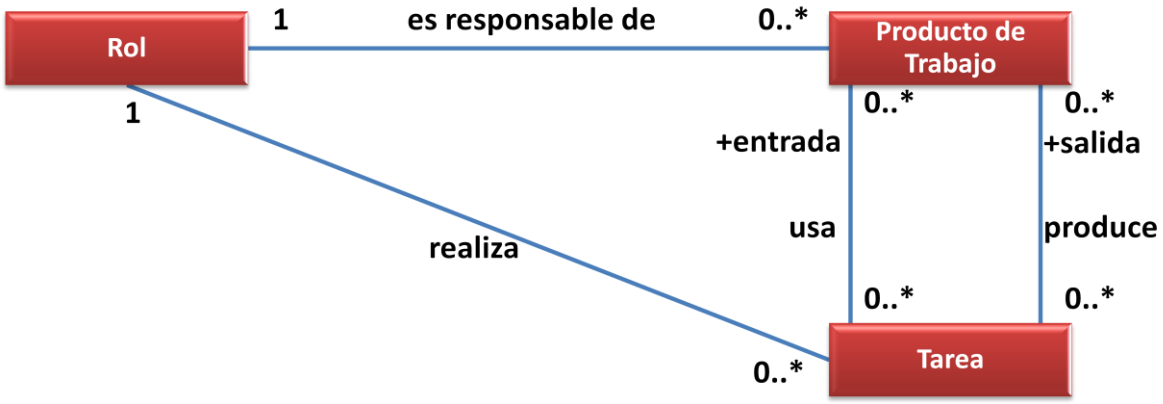



**Figura 2.** Idea básica de proceso en SPEM 2.

### 2.5.1 El Meta Modelo SPEM

El Meta Modelo SPEM está formado por siete (7) paquetes o unidades lógicas principales (ver Figura 3). Cada paquete extiende a la unidad o paquetes de los cuales depende, adicionando estructuras y capacidades.

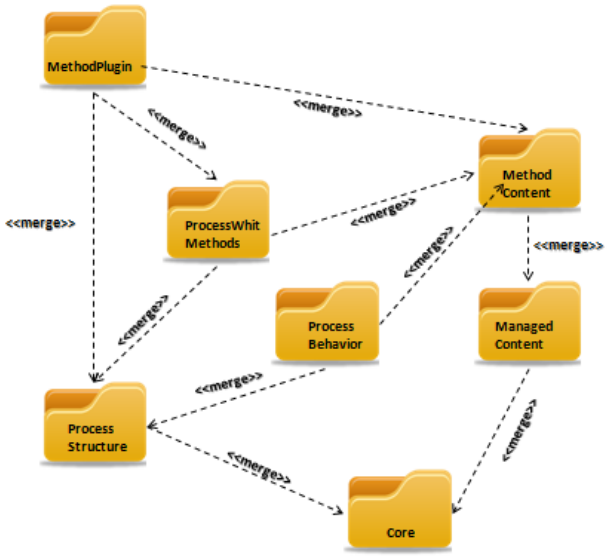



**Figura 3.** Estructura de paquetes de SPEM 2. Tomado de [48].

Un ingeniero de procesos puede utilizar diferentes niveles de capacidad, conjuntos de conceptos, y niveles de formalismo para expresar sus procesos utilizando unos u otros

paquetes. En nuestro caso enfatizaremos en el paquete *Method Content* (Contenido de método) con el fin aplicar algunas métricas de acoplamiento y cohesión, para poder hacer un análisis visual sobre su resultado.

### 2.5.2 Contenido de Método (Method Content)

Contiene los conceptos para elaborar elementos de métodos, que son los ítems básicos que sirven de base para el ensamblado de procesos, metodologías, ciclos de vida, etc. Los principales elementos de método se obtienen de la premisa de SPEM: alguien (rol) hace algo (tarea) para obtener algo (producto de trabajo) basándose o ayudándose en algo (guía). Por lo anterior, los elementos de método permiten describir, con el detalle necesario, cómo se alcanzan los objetivos del proceso haciendo qué tareas por qué roles usando qué recursos y obteniendo qué resultados. El contenido de método puede ser organizado mediante una jerarquía de paquetes de contenido (Content Package), cada uno de los cuales puede incluir roles, tareas, productos de trabajo y guías, considerados también patrones primitivos de trabajo. En consecuencia, los cuatro tipos de elementos de contenido son: Tarea, Rol, Producto de Trabajo y Guía.

La Figura 4 presenta la jerarquía de los conceptos empleados en el Method Content, incluidos los elementos de contenido (Content Element). Los conceptos de Contenido de Método se muestran organizados jerárquicamente, así mismo se incluyen los elementos de contenido (Content Element) que son: Tarea, Rol, Producto de Trabajo y Guía.

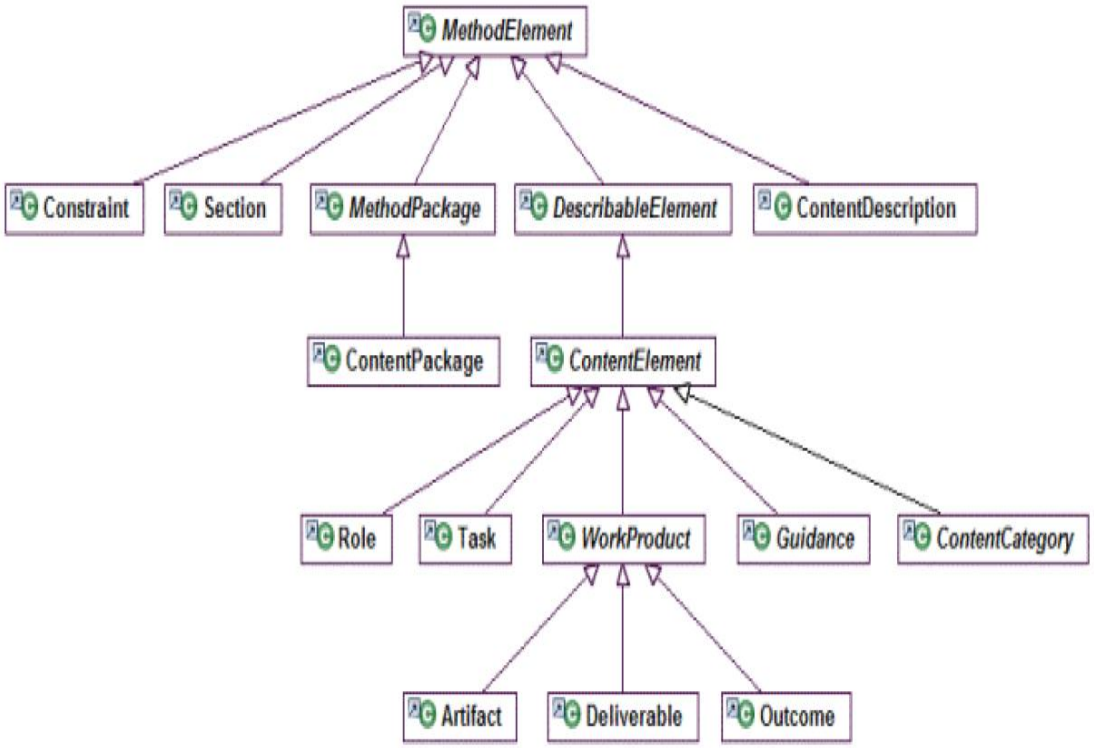



**Figura 4.** Jerarquía de conceptos del method content. Tomado de [48].

## 2.6 Proyecto Eclipse Process Framework

Eclipse[10] es una comunidad de código abierto en la que sus proyectos están orientados a la elaboración de una plataforma de desarrollo abierta conformada de entornos, herramientas y plugins extensibles, para la construcción implantación y gestión de software a través de todos su ciclo de vida. Eclipse Process Framework[11] (EPF) tiene como objetivo proporcionar soluciones a los problemas más comunes a los cuales se enfrentan gerentes y equipos de desarrollo al momento de adquirir y administrar sus métodos y procesos. EPF permite soportar, a través de una plataforma configurable una extensa variedad de tipos de proyectos y tipos de desarrollo. Los objetivos del proyecto EPF son:

- Proveer un marco extensible (y herramientas de ejemplo) para la ingeniería de procesos de software, en las tareas de creación, gestión, configuración y publicación de procesos.
- Proveer modelos de procesos extensibles, para la administración y desarrollo de software incremental, ágil e iterativo, aplicables a un amplio conjunto de plataformas de desarrollo y aplicaciones.

## 2.7 La Aplicación EPF Composer

Eclipse Process Framework Composer[12] (EPF Composer) es una plataforma de software gratuita nacida de la iniciativa EPF generada para ingenieros de procesos, jefes de proyecto y directores de programas que se encargan de mantener e implementar los procesos de las organizaciones de desarrollo o proyectos individuales, esta plataforma permite especificar formalmente procesos de desarrollo de software, para ello, cuenta con herramientas para la creación, configuración, visualización y publicación de métodos y procesos de desarrollo de software estructurados en un esquema predefinido. Este esquema es una evolución de la versión 1.1 de SPEM, aunque actualmente el equipo de EPF Composer soporta la versión 2.0 de SPEM.

## 2.8 Moose: Plataforma de Análisis Visual

Moose[13] es una plataforma de código abierto para la expresión de los análisis de los sistemas de software y de los datos en general, su objetivo principal es ayudar a los

[10] http://www.eclipse.org/
[11] http://www.eclipse.org/epf/
[12] http://www.eclipse.org/epf/tool_component/tool_vision.php
[13] http://www.moosetechnology.org/

ingenieros a comprender grandes cantidades de datos, y los sistemas de software en particular. Desde un punto de vista conceptual, Moose se organiza de la siguiente manera, ver Figura 5.

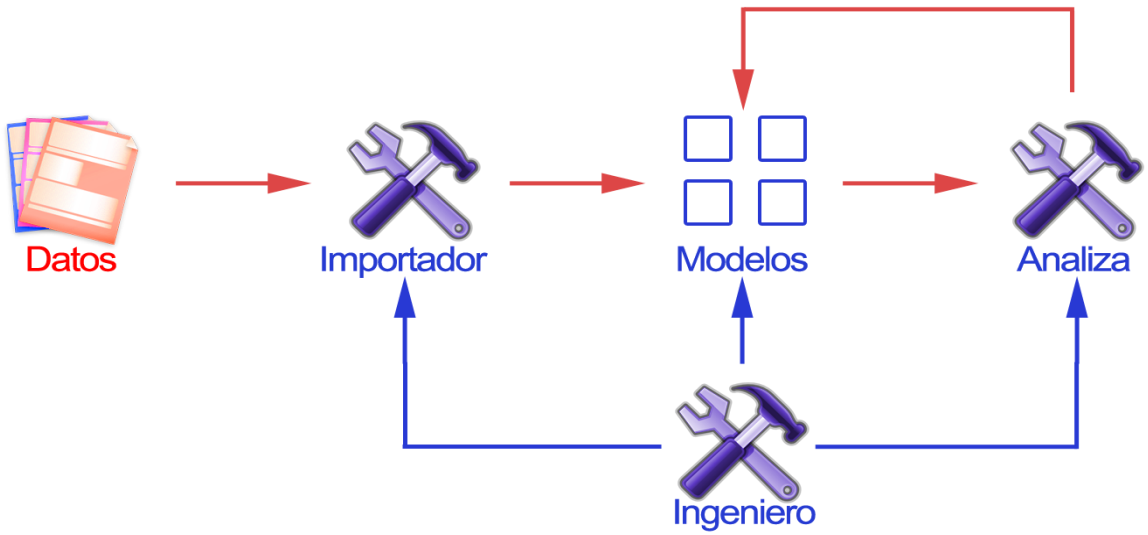



**Figura 5 . Flujo de trabajo general de Moose, adaptado de [49]**

Moose se dirige a varias categorías de personas:

- Investigadores en el área de ingeniería de análisis de software, minería de datos e ingeniería inversa.
- Ingenieros y arquitectos que quieren entender los sistemas y datos.
- Constructores/Desarrolladores de herramientas.

El insumo o entrada para el motor de Moose son datos. Por datos se entiende que son todo los tipos de estructuras que contienen objetos, propiedades y relaciones. Por ejemplo, un sistema de software escrito en Java, o bien, un conjunto de archivos de configuración escritos en XML, entre otros. Estos datos se cargan en Moose través de importadores. En Moose, se puede importar datos de diversas fuentes y en diferentes formatos. Por ejemplo, Moose puede manejar la importación de la estructura de los sistemas de software escritos en varios lenguajes de programación ya sea a través de importadores internos (por ejemplo, Smalltalk[14], XML, MSE), o a través de las externas (por ejemplo, Java, JEE, C + +). Una vez importado, los datos se almacenan en los modelos con los cuales se pueden comenzar a realizar distintos tipos de análisis por parte de los ingenieros. Moose permite manipular grandes cantidades de datos, pero presenta un problema al momento de comprender y evaluar su estado, ya que los datos no tienen una forma física o visual entendible. La falta de forma gráfica de los datos hace inútil la habilidad humana para interpretar los modelos por medio de estímulos visuales. Mondrian, es un motor de scripting basado en Smalltalk que permite diseñar

[14] http://www.squeak.org/Smalltalk/

visualizaciones personalizadas de datos en una manera rápida dentro de la plataforma Moose [49].

## 2.9 Pharo: Entorno de programación.

Pharo[15] es un entono de programación de código abierto para el desarrollo de software profesional, con todas las funciones del lenguaje de programación Smalltalk. Pharo es altamente portátil, incluso su máquina virtual está escrita completamente en Smalltalk, por lo que es fácil de depurar, analizar y cambiar.

El ambiente de desarrollo Smalltalk es principalmente gráfico y funciona como un sistema en tiempo de ejecución que integra varias herramientas de programación (Smalltalk), y servicios del sistema operativo. En Smalltalk se manipula el entorno mismo, comúnmente mediante el navegador del Sistema. La plataforma Pharo es una simulación de un mundo habitado por objetos, que se comunican entre sí a través de mensajes contenidos en guiones de acción. La metáfora del mundo hace de éste el escenario en el que transcurre toda la acción. La plataforma Moose trabaja sobre Pharo.

## 2.10 Métricas asociadas a modularidad de software

Es de gran utilidad diseñar sistemas estables y con soporte al constante cambio, ya que, sistemas software que han tenido éxito en la industria, frecuentemente tienen una larga vida, porque una vez que el proceso ha demostrado su utilidad, es importante modificarlo para dar cabida a los cambios y con ello añadir funcionalidad para aumentar su utilidad. Pero desafortunadamente, no todos los sistemas software se han diseñado pensando en la modularidad, lo cual dificulta la reutilización. Lehman[50], afirma que cuando se realizan cambios en el software, junto con la creciente interconexiones entre los elementos del sistema, conduce al aumento de la complejidad del sistema y la corrupción de la estructura de software.

La tendencia de la ingeniería de software consiste en introducir la medición en todas las etapas del ciclo de vida de un sistema, independientemente del paradigma utilizado. En este contexto, las métricas de software proporcionan fórmulas matemáticas para medir diferentes atributos del software. Entre los más frecuentes se encuentran el tamaño, la complejidad, la frecuencia esperada de aparición de errores, el grado de acoplamiento y la cohesión, entre otros. Buscando medir atributos, que ayuden al desarrollo, y mantenimiento del software principalmente.

[15] http://www.pharo-project.org

### 2.10.1 Modularidad

La modularidad es un atributo interno de calidad que influye en las características externas de calidad de software [51]. La modularidad puede ser considerada como un principio fundamental de la ingeniería, ya que facilita, entre otras cosas:

- Diseñar y desarrollar diferentes partes del mismo sistema por diferentes personas, a menudo pertenecientes a diferentes empresas.
- Permite manejar la complejidad de los sistemas mediante el fraccionamiento de las partes grandes de acoplamiento, las cuales pueden ser entendidas de una mejor manera.
- Brinda la posibilidad de poner a prueba el sistema de forma paralela (a diferentes personas al mismo tiempo).
- Puede sustituir o reparar diferentes partes del sistema sin afectar otras partes del mismo
- La reutilización de partes existentes en diferentes contextos.
- Para dividir el sistema en unidades de configuración los cuales pueden ponerse bajo el control de la configuración.
- Para limitar la propagación de cambios.

La modularidad ha sido aplicada para analizar el diseño de modularidad de sistemas orientados a objetos, ese nivel de abstracción un modulo es un conjunto de clases. En donde se usa la información de las dependencias entre las clases para mejorar la modularidad. Una buena modularización de un sistema permite que realizar cambios en partes del sistema sin afectar otras partes del sistema que no guarden relación. Una buena organización de clases en paquetes cooperativos identificables hacen que el entendimiento y las actividades de mantenimiento, pruebas y evolución del software se realicen sin mayores inconvenientes [52]. Sin embargo, una vez se ha modularizado correctamente un sistema software, su estructura decae, debido a que el software evoluciona a través del tiempo con la modificación, adición y eliminación de clases y las dependencias entre clases. En consecuencia, la modularización progresiva se erosiona y pierde calidad [53]. Para mejorar la modularidad de los paquetes es necesario realizar una evaluación de la organización y relaciones establecidas entre los mismos.

### 2.10.2 Acoplamiento y Cohesión

La cohesión y el acoplamiento permiten evaluar y mejorar la programación y el diseño de sistemas informáticos, aplicaciones, modelos de software, entre otros. Marcus y Ferenc [51], definen la cohesión como “la medida de dependencia entre los diferentes elementos de un modulo” y el acoplamiento como “la medida de la fuerza de conexión de un módulo a otro”. En el diseño estructurado, se obtiene alta cohesión cuando cada módulo (función o procedimiento) realiza una única tarea trabajando sobre una sola estructura de datos [54]. Expertos en ingeniería de software, han manifestado que la alta cohesión y el bajo acoplamiento son los principales factores para identificar la modularidad [51]. En

lenguajes orientados a objetos como Java, Smalltalk y C++, la estructura de paquetes ha permitido a las personas organizar sus programas en subsistemas. Por lo tanto, mientras más fuerte sea el acoplamiento y entre más baja sea la cohesión, los modelos desarrollados son más difíciles de entender, cambiar, reutilizar, y corregir. Dado que estos conceptos resultan útiles para el análisis de software, éste trabajo de grado busca aplicarlos al análisis de modelos de procesos de software con el fin de ayudar a analizar visualmente si la modularidad ha sido bien lograda.

Muchas de las tareas de mantenimiento requieren que el desarrollador identifique puntos de cambio y realice los cambios en esos puntos. El acoplamiento es una de las propiedades con más influencia en el mantenimiento, ya que tiene un efecto directo sobre mantenibilidad[6] al identificar la dependencia entre los módulos y por tanto el impacto que tiene el cambio de un módulo sobre los demás. En la programación orientada a objetos se dice que dos clases están acopladas cuando los métodos declarados en una clase, usan los métodos o variables de instancia definidos por otra clase. Un alto acoplamiento entre clases, significa que éstas dependen mucho unas de otras y por lo tanto resulta difícil tanto la reutilización en otras aplicaciones, como el mantenimiento del sistema y por ende el mantenimiento resulta ser costoso [32] [35].

La cohesión en modelos de procesos de software se centra en la fuerza con las tareas, roles, o artefactos del proceso de un paquete de contenido de método están relacionados con, o conectado, el uno al otro y el acoplamiento se refiere a la independencia entre los paquetes de contenido de método para ser modificados y por lo tanto, la menor relación posible entre las partes.

#### 2.10.2.1 Acoplamiento aferente, acoplamiento eferente e inestabilidad.

Las métricas asociadas al acoplamiento aferente (hacia dentro) y acoplamiento eferente (hacia afuera) están relacionadas con el cálculo de un paquete y la relación de los mismos, lo que da lugar a una métrica denominada inestabilidad de un paquete [52]. Un componente A tiene una relación aferente con un componente B, si un atributo o método de B está vinculado a un método o atributo de A, lo que significa que B requiere los servicios de A, y por lo tanto, depende de A. En ese sentido, el cálculo del acoplamiento aferente es el resultado de la suma de las clases, atributos o métodos que tiene B y dependen de A [55] [56]. Un componente A tiene una relación eferente con un componente B, si un atributo y/o método provisto de B está vinculado a un atributo y/o método de A, lo que significa que B proporciona servicios hacia A y, por lo tanto, se A depende por B, por lo tanto, el acoplamiento eferente se obtiene como la suma de clases de otros paquetes de los cuales dependen clases del paquete con el que se está trabajando [55] [56]. La inestabilidad de un paquete es la métrica que enfrenta entre sí al acoplamiento aferente y eferente a través de la siguiente fórmula [55] [56] :

> *Inestabilidad = Acoplamiento eferente / (Acoplamiento eferente + Acoplamiento aferente)*

Por tanto los valores de inestabilidad se encontrarán dentro del rango de valores entre 0 y 1.

## 2.11 Trabajos relacionados

La calidad del modelo de proceso de software ha sido analizado usando enfoques tradicionales, tales como: métricas [11], pruebas [8] [9], simulaciones [10] y verificaciones formales[12].

### 2.11.1 Enfoques tradicionales para el análisis de procesos

El mantenimiento del proceso de software merece la misma atención que otro tipo de software [57]. Por lo anterior es necesario evaluar la capacidad de mantenimiento de los procesos en las primeras etapas de su desarrollo, principalmente durante el proceso de modelado. La mantenibilidad de software ha sido ampliamente tratada en la literatura y ha habido varios esfuerzos en los últimos años para caracterizar y cuantificar la misma. Una gran variedad de trabajos han estudiado la mantenibilidad del software del producto como una variable dependiente y la mayoría de ellos se han enfocado en el código. En el paradigma de orientación a objetos Harrison et. al [58] y Briand et. al [59] han desarrollado modelos de predicción para las tareas de mantenimiento. Varios autores han investigado la evaluación de la capacidad de mantenimiento en las primeras etapas del desarrollo de producto de software. Algunos trabajos importantes relacionados con el estudio de la mantenibilidad de modelos UML son:

- Marchesi [60] y Saeki [61] proponen métricas para los diagramas de casos de uso.
- Genero et. al [62] presentan un conjunto de métricas para los diagramas de clases.
- Miranda et. al [63] definen métricas para los diagramas de gráficas de estado (statechart).

Canfora et. al. [64], definen un conjunto de métricas para medir de manera global un proceso de software, sin embargo las mediciones son realizadas sobre todo el proceso de desarrollo y como una sola entidad, sin embargo en un análisis de procesos basado en esos datos generales, es difícil conocer cuál es la cohesión de las partes y el acoplamiento entre éstas. Sin embargo, estas métricas son un punto de partida interesante puesto que logran tener resultados relevantes para evaluar la usabilidad y mantenibilidad del modelo de proceso, aunque no es concluyente respecto a la mantenibilidad. En nuestro trabajo de grado usamos las métricas relacionadas a la modularidad de software con el fin de evaluar el acoplamiento, cohesión e inestabilidad y de esta forma construir una vista que permita validar aspectos más específicos de los diferentes elementos del modelo de proceso. En nuestro trabajo definimos un enfoque de

evaluación basado en la visualización de paquetes de método y su contendido relacionado a roles, tareas y productos de trabado, mientras que Canfora et. al proponen y validan algunas métricas a nivel general. Nuestro trabajo se enfoca en cómo la visualización de un modelo de proceso permite realizar un análisis de los datos y detectar problemas [65]. Además de visualizar los diferentes problemas en el modelo de proceso, se utiliza la capacidad humana para interpretar las vistas de los modelos de procesos, con el fin que se logre evaluar la coherencia y confiabilidad de un proceso de software antes de ponerlo a prueba en proyectos reales, y con ello para mitigar posibles inconvenientes en su aplicación.

Por otro lado, en el enfoque de pruebas para los Modelos de Procesos, se toman los resultados o evidencias de la ejecución del modelo aplicado a un proyecto, y se comparan con las especificaciones de un modelo de calidad [20]. Un ejemplo de pruebas de proceso es el modelo de evaluación conducida, donde un modelo de proceso es evaluado con un framework de capacidad de madurez, tales como el enfoque de CMMI e ISO/IEC15504, pero este tipo de actividades de prueba sólo pueden llevarse a cabo una vez que el modelo de proceso ya se ha aplicado a proyectos específicos. Así, las pruebas del modelo de proceso necesitan ciclos muy largos, ya que se debe verificar el cumplimiento o adaptación al estándar, y no se garantiza la calidad del modelo de proceso mismo.

Gruhn [10], ha propuesto una técnica de verificación basada en los resultados de simulación y evaluación de la traza de ejecución de los procesos. Teniendo en cuenta que el análisis de modelos de procesos de software es un tema prometedor, entonces se puede estar seguro de que los problemas detectados en un modelo de proceso afectan eventualmente el proceso de software como tal. Análogamente, se puede asegurar que, si un modelo de proceso M ha demostrado que está bien elaborado (p ej. ausencia de bloqueos, número limitado de objetos de cierto tipo), entonces todos los procesos de software $PS_1,\ldots,PS_n$ administrados por el modelo ($PS_1,\ldots,PS_n \in PS(M)$) tendrían la misma propiedad de estar bien elaborados. Por lo tanto, el análisis de modelos de proceso de software ayuda a prevenir la ejecución de errores en los modelos de proceso de software. Así, el análisis de modelos de software ayuda a disminuir los recursos, mano de obra y de la misma forma recursos hardware. La principal debilidad existente en la técnica de validación propuesta por Gruhn, es que no todos los resultados son validos para los procesos de software pertenecientes al modelo de proceso ($PS_1,\ldots,PS_n \in PS(M)$). Cada resultado de la validación es sustentado solo por experimentos. Por tanto la simulación tiene un ciclo más corto, pero aún requiere de gran cantidad de datos históricos, además esta técnica de verificación es adecuada sólo cuando se sabe que el modelo de proceso es conveniente para la organización o el proyecto, pero si el modelo está incompleto, insuficientemente especificado o no es conveniente, la simulación no producirá los resultados esperados. Con el enfoque de AVISPA que sigue este proyecto, los modelos de proceso podrán ser analizados aún si estos son incompletos, por otro lado, la presencia de las inconsistencias es lo que le da sentido a este tipo de análisis.

Christov y Avrunin [66], presentan una técnica formal de verificación y validación para identificar y medir las diferencias entre los modelos de procesos y las ejecuciones reales. Pero no se enfocan en la integridad o la coherencia del modelo de proceso, sino que evalúan la congruencia entre el modelo y en el entorno del proyecto, basándose en datos históricos. Sin embargo, la obtención de estas mediciones no es fácil y los resultados no son precisos [67]. Osterweil [20] afirma que los procesos de software son también software y por lo tanto las técnicas que se aplican al software, también pueden ser aplicadas en modelos de procesos, por ejemplo las inspecciones de software, recorridos y revisiones técnicas, son usadas ampliamente para mejorar la calidad del software, por lo que también son técnicas prometedoras para los procesos de software. Babar [68], presenta un enfoque donde la idea principal es reunir un grupo de ingenieros de software con experiencia, para revisar un producto software, con el fin de mejorar la calidad, mediante la detección de defectos que el diseñador o desarrollador del producto podrían haber introducido o pasado por alto. En éste sentido, la idea de la revisión, se basa en un proceso de revisión, el cual contiene tres tareas principales: descubrimiento del defecto, recopilación de defectos, y la discriminación defecto, en donde el descubrimiento del defecto es el que exige la más alta experiencia. El enfoque y la herramienta que se presenta en éste trabajo de grado tiene por objetivo facilitar la identificación de defectos en el modelo de proceso de software que lo hacen menos mantenible, usando un enfoque intuitivo y en un contexto de revisión. Además, la solución específica que se presenta en este documento puede ser útil para un diseñador de procesos para obtener algunas ideas de mejora sobre el diseño del modelo del proceso y potencialmente encontrar algunos problemas.

### 2.11.2 Enfoques basados en visualización para el análisis de procesos

AVISPA [14], es una herramienta que construye Blueprints y resalta los patrones de error. Se enfoca sobre los elementos de contenido: Rol, Tarea y Producto de trabajo; y con ello especializa los Blueprint a un tipo de elemento a la vez. Sin embargo AVISPA no incluye patrones para evaluar la modularidad de un modelo de proceso. Éste trabajo de grado, propone generar Blueprints para el análisis visual del acoplamiento y cohesión de paquetes de método y paquetes de proceso como una extensión a AVISPA. La idea es definir un modelo de métricas que faciliten mostrar en una forma visual algunos patrones de error referentes al acoplamiento y cohesión de los paquetes, ya que un gran acoplamiento y una baja cohesión pueden ser desventajosos a la hora de corregir errores, integrar partes, o hacer mantenimientos. Existe una gran cantidad de trabajos enfocados en la visualización de las clases y los métodos [49], la arquitectura de software [69], e incluso los comentarios del código [70], en donde la búsqueda de patrones de error en el código fuente ha sido bastante exitosa [71] [72], así que después de un enfoque similar en AVISPA, se ha logrado, luego de una vasta experiencia empírica, identificar y localizar de forma automática una serie de patrones de errores en los modelos de procesos de software.

En este proyecto se propone usar métricas como base para la construcción de capas de visualización que ayudan a los ingenieros de procesos a localizar problemas en los modelos, logrando agrupar Rol, Tarea, Productos de Trabajo. Se propone un enfoque complementario a AVISPA, para analizar los modelos de manera temprana, con base en la revisión de las vistas de arquitectura de los modelos de procesos de software definidos como Blueprints de Procesos de Software [1]. Allí, cada Blueprint es construido siguiendo una estrategia dirigida por modelos donde se separa el modelo de proceso en un conjunto de vistas parciales que pueden ser más ilustrativos para encontrar errores. Sin embargo, la usabilidad y la complejidad de los Blueprint se ve amenazada cuando se trata de modelos de procesos grandes y complejos. Al identificar un conjunto de patrones de error, y ponerlos en evidencia, se logra encapsular el conocimiento especializado de un ingeniero experto en procesos de software para la identificación de oportunidades de mejora y que requieren por tanto, menos ingenieros de proceso. El uso de la visualización para identificar los patrones de error no es nuevo, anteriormente, grupos de investigadores utilizaron esta técnica para identificar patrones visuales, positivos o negativos de los sistemas de software [67] [73] [74].

# 3 ANÁLISIS VISUAL DE LA MODULARIDAD DE LOS MODELOS DE PROCESO

## 3.1 Introducción al modelo de visualización

La Figura 6 presenta un modelo arquitectónico sobre los diferentes niveles de abstracción de un modelo de proceso de software en AVIMO-PS, manteniendo la arquitectura definida en AVISPA. En el primer nivel están los modelos de proceso especificados en el lenguaje SPEM2.0 en EPFC. Estos procesos son importados por AVIMO-PS considerando los paquetes de métodos y sobre ellos se aplican un conjunto de métricas relacionadas con la modularidad de dichos paquetes de método, las cuales son calculadas de acuerdo a fórmulas bien definidas. Basado en éstas métricas, un conjunto de Blueprints de modularidad son desplegados y sobre éstos algunos patrones problemáticos son resaltados a través de colores, para ayudar a la evaluación visual del modelo del proceso de software. A continuación se describen cada uno de éstos niveles arquitectónicos de AVISPA y las implicaciones particulares para AVIMO-SP.


**Figura 6. Modelo de AVISPA en la localización de oportunidades de mejora de modelos de procesoss de software[75]**

## 3.2 Modelos de Proceso SPEM 2.0

AVIMO-PS usa modelos SPEM2.0 de manera similar a AVISPA, en la cual los modelos de proceso son importados desde un archivo XML, generados por la herramienta Eclipse Process Framework Composer - EPFC. Sin embargo, AVIMO extiende la capacidad de análisis de AVISPA recuperando desde EPFC la información necesaria para evaluar aspectos de modularidad en los modelos de proceso. La Figura 7, presenta una estructura de descomposición de una librería de método SPEM 2.0 deducida de la estructuración de un modelo de proceso en SPEM2.0. Esta estructura de descomposición muestra que la forma de modularizar un proceso SPEM 2.0 es través de paquetes, bien sea de contenido de método o de proceso.

Los paquetes de proceso definen cuándo las tareas se llevan a cabo a través de diagramas de actividad y/o estructuras de desglose de trabajo; estos paquetes albergan los procesos de software y sus descomposiciones, además son útiles al momento de reutilizar fragmentos de proceso y configurar distintas soluciones específicas para organizaciones y proyectos. Por otro lado, los paquetes de contenido de método, son los que finalmente albergan el conocimiento de los roles, productos de trabajo y tareas que son articulados en los proyectos; además contienen básicamente el capital intelectual de una compañía y forma la base del conocimiento para describir cómo se alcanzan las metas a nivel de desarrollo, dicho de otra manera almacena los elementos que rigen el comportamiento del proceso. Una vez definidos los elementos de contenido del método (tareas, roles y artefactos), éstos son reutilizados para crear procesos. Aunque estos elementos se pueden definir directamente al crear un proceso, esto dificulta mucho la reutilización y actualización de dichos elementos. Por lo anterior, se ha optado por crear primero los elementos y reutilizarlos después para crear los procesos

Al momento de realizar el mantenimiento de un proceso, es importante mantener tanto los flujos de trabajo, como el contenido de método. Normalmente los procesos de software no cuentan con un número considerable de paquetes de procesos y por tanto, éstos no exhiben una necesidad grande de lograr una adecuada modularidad. Sin embargo, los paquetes de contenido de método, son los que finalmente agrupan gran cantidad de activos de proceso a través de definiciones de los elementos básicos de SPEM 2.0 como lo son: roles, tareas, productos de trabajo y las guías. AVIMO-PS por tanto, centra su procesamiento en los paquetes de contenido de método de SPEM 2.0 para facilitar el análisis de la modularidad de modelos de proceso. Para ello AVIMO-PS ha extendido la funcionalidad del importador de procesos de AVISPA para poder leer información de los paquetes de contenido de método y resolver las referencias con respecto a los elementos del modelo de proceso ya incorporados por AVISPA (tareas, roles y artefactos).

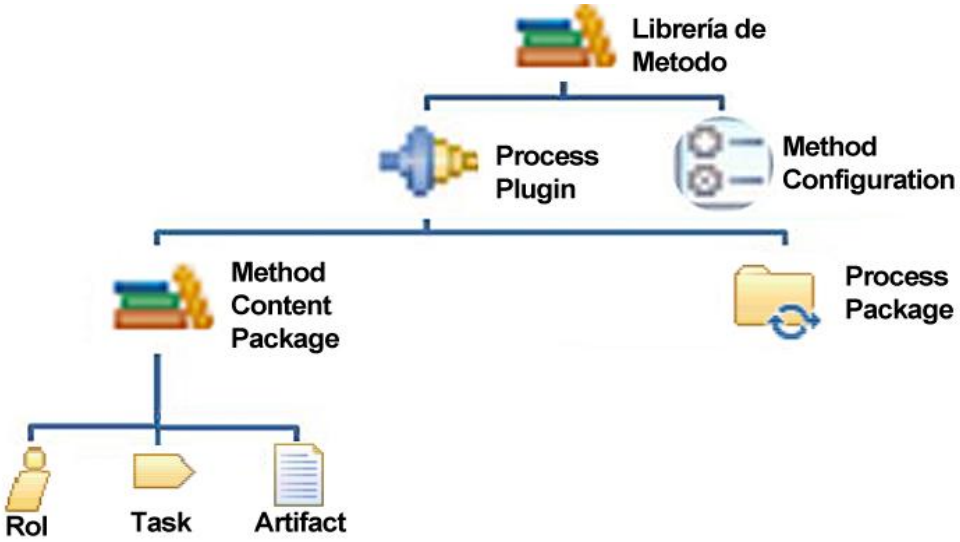



**Figura 7. Estructura de composición de los elementos SPEM 2.0**

## 3.3 Métricas de modularidad

La modularidad es una partición lógica del diseño software que permite al software complejo ser manejable para propósitos de implementación y mantenimiento[76], por esta razón, se han realizado numerosos estudios al rededor de la modularidad, enfocados en la obtención de métricas que ayuden a medirla de una manera más directa, los cuales han llegado a un consenso dentro del campo de las métricas de software orientado a objetos, al señalar a las métricas de acoplamiento y cohesión como un indicador confiable de la calidad de un programa orientado a objetos [23]. Del mismo modo muchos autores han estudiado la modularidad, resaltando que los sistemas modulares son fáciles de entender y explicar porque se puede estudiar cada módulo por separado, y además porque estos módulos tienen bien definidas las interdependencias, gracias a que son altamente cohesivos y débilmente acoplados [77], [32], [78]. De aquí, expertos en ingeniería de software aseguran que los diseños con bajo acoplamiento y alta cohesión dan lugar a productos que son a la vez, más fiables y más fáciles de mantener [6] [23]. Por otro lado Yourdon[79] y McCabe[80], proponen que además del acoplamiento y cohesión, existe la inestabilidad del sistema, la cual afecta de forma negativa la modularidad, afirmando que la identificación y caracterización de zonas inestables en el sistema puede ayudar en gran medida al mantenimiento, y permitir en tiempo de diseño anticipar y gestionar los cambios futuros. Del mismo modo, la inestabilidad puede indicar dificultad del sistema para adaptarse a un entorno cambiante. Es por ello que se logra destacar el acoplamiento, cohesión e inestabilidad como las métricas que más peso y significado tienen sobre la modularidad de los sistemas software.

Osterweil[20], expone que los procesos de software también pueden ser considerados como software, lo cual es un punto de apoyo para centrar este trabajo de grado, en la

adaptación de métricas de modularidad de modelos de procesos de software, tomando como punto de partida las métricas de acoplamiento, cohesión e inestabilidad definidas en la programación orientada a objetos.

Es por ello que a los modelos de proceso de software que se importan a AVIMO-PS, se aplican un conjunto de métricas que han sido adaptadas de la literatura asociada a la medición de la modularidad en software, al encajar con el modelo conceptual de SPEM2.0 y que permitieran obtener visualizaciones específicas. Las métricas que se han definido para facilitar el análisis de la modularidad de los paquetes de contenido de método son: cohesión, acoplamiento e inestabilidad. A continuación se describen conceptualmente éstas métricas y se definen las fórmulas establecidas para su cálculo.

### 3.3.1 Nivel de cohesión de los paquetes de contenido de método

Lo ideal en los paquetes de contenido de método, es diseñar el proceso de tal manera que sus elementos trabajen conjuntamente para alcanzar un propósito único y bien definido. De esta manera la cohesión del paquete se determinó examinando el grado en que el conjunto de elementos de método que éste incluye, estén altamente relacionados. Por tanto la métrica de cohesión corresponde a la proporción entre el conjunto de elementos cohesivos del paquete (para los cuales la suma de sus relaciones internas al paquete que lo contiene es mayor que las externas a otros paquetes) y el número total de elementos. El rango de la función de cohesión está entre 0 y 1, donde en un módulo con alta cohesión, ésta debería estar cercana a 1, indicando que mantiene un propósito bien definido en el paquete.

$$\mathbf{Cohesion}_{\mathbf{Pi}} = \frac{|\mathrm{PRi}|}{|\mathrm{Pi}|}$$

**Fórmula 1. Cohesión**

Donde:
PRi es número de elementos dentro del paquete i-ésimo, que cuentan con más relaciones internas que externas dentro del Paquete.
Pi, es el número total de los elementos dentro del paquete i-ésimo.

La Figura 8, muestra las conexiones entre las tareas de los paquetes “Monitoreo y control del proyecto” y “Desarrollo de Requerimientos”, aplicando la métrica de Cohesión en los paquetes, obtenemos:

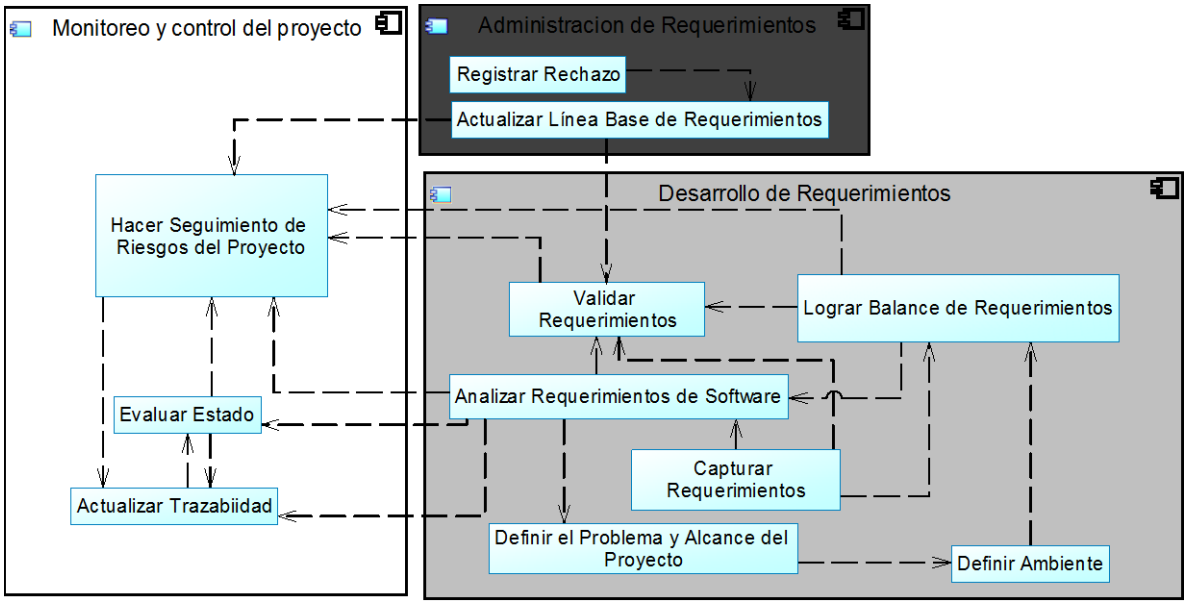



**Figura 8. Cohesión de paquetes**

- $\boldsymbol{Cohesion}_{\mathbf{MonitoreoControl}} = \frac{3}{3} = 1$, este valor indica que el paquete "Monitoreo y control del proyecto" tiene un nivel de cohesión ideal, porque sus elementos están totalmente relacionados entre sí.
- $\boldsymbol{Cohesion}_{\mathbf{DesarrolloRequerimientos}} = \frac{4}{6} = 0.66$, este valor indica que el paquete "Desarrollo de Requerimientos" tiene el 66.8% de elementos presenta la mayoría de sus relaciones con elementos fuera del paquete y por tanto cabe la posibilidad de que muchas de ellas no hagan parte de los intereses de paquete.
- $\boldsymbol{Cohesion}_{\mathbf{AdministracionRequerimientos}} = \frac{1}{2} = 0.5$, este valor indica que el 50% de los elementos del paquete "Administración de Requerimientos", tienen la mayoría de sus relaciones con tareas fuera del paquete, reduciendo drásticamente la cohesión o especialización del paquete, se entiende que paquetes más especializados son más mantenibles, ya que una baja cohesión facilita el hecho de que los paquetes realicen muchas actividades y/o acciones diferentes.

Nuestra propuesta soporta tres maneras de analizar la cohesión de los paquetes de contenido de método, de acuerdo a la relación entre elementos del mismo tipo así:

- Cohesión del paquete a nivel de tareas
- Cohesión del paquete a nivel de roles
- Cohesión del paquete a nivel de artefactos

### 3.3.2 Métricas de acoplamiento entre paquetes de contenido de método

El acoplamiento en los paquetes de método, hace referencia al grado de dependencia que tienen los elementos en los paquetes del modelo, el cual se calcula, utilizando el

Acoplamiento Eferente (hacia afuera), consultando los elementos sucesores de un elemento i que se encuentran en otro paquete. Por lo tanto, si se considera que hay n elementos en un paquete (p), el acoplamiento dependiendo del tipo de elemento, se calcula de esta manera:

$$\mathbf{Ae_p} = \sum_{i=0}^{n} \mathbf{Elemento_i}.\,\text{relacionesSalientesAOtrosPaquetes}, \forall\, Elemento \in p$$

**Fórmula 2. Acoplamiento de cualquier elemento en paquetes**

En particular, aplicando la formula de la métrica a cada elemento se tiene:

$$\mathbf{At_p} = \sum_{i=0}^{n} \mathbf{Tarea_i}.\,\text{relacionesSalientesAOtrosPaquetes}, \forall\, Tarea \in p$$

**Fórmula 3. Acoplamiento de tareas en paquetes**

$$\mathbf{Ar_p} = \sum_{i=0}^{n} \mathbf{Rol_i}.\,\text{relacionesSalientesAOtrosPaquetes}, \forall\, Rol \in p$$

**Fórmula 4. Acoplamiento de roles en paquetes**

$$\mathbf{Aa_p} = \sum_{i=0}^{n} \mathbf{Artefacto_i}.\,\mathbf{relacionesSalientesAOtrosPaquetes}, \forall\, \boldsymbol{Artefacto} \in \boldsymbol{p}$$

**Fórmula 5. Acoplamiento de artefactos en paquetes**

**Rango**: [0…∞)

La Figura 9 muestra las relaciones entre los roles de los paquetes “Pruebas”, “Comercial” y “Administración”, aplicando la métrica de Acoplamiento en los paquetes, se obtiene:

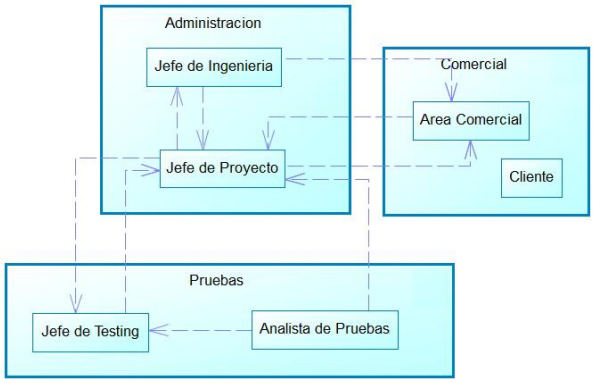



**Figura 9. Acoplamiento de roles en paquetes**

- $\mathbf{At_{Administracion} = 3,}$ éste valor indica el grado de dependencia hacia otros roles, indicando que el paquete depende en tres oportunidades de otros roles que están en otros paquetes: dos (2) para el “Área Comercial” y una (1) para “Jefe de Testing”

- $\mathbf{At_{Pruebas} = 2,}$ éste valor indica el grado de dependencia hacia otros roles, indicando que el paquete depende en dos oportunidades de un rol que está en otro paquete: “Jefe de Proyecto”.

- $\mathbf{At_{Comercial} = 1}$, éste valor indica el grado de dependencia hacia otros roles, indicando que el paquete depende en una oportunidad de un rol que está en otro paquete: “Jefe de Proyecto”.

Como se muestra en las siguientes secciones, el resultado de esta métrica se utilizará para generar los Blueprints de acoplamiento y cohesión, e inestabilidad, y con ello mostrar gráficamente la intensidad del acoplamiento entre paquetes de contenido de método, mediante una línea con punta entre los paquetes.

### 3.3.3 Inestabilidad en paquetes de contenido de método

Otra métrica de calidad que se propone en este trabajo de grado, que está relacionada con el cálculo de acoplamiento (hacia afuera) y el acoplamiento aferente (hacia adentro) de un paquete y la relación entre los mismos, que da a lugar a la métrica de inestabilidad de un paquete de contenido de método. Estas tres métricas son una adaptación de las fórmulas que propone Robert Cecil Martin[81].
La combinación del acoplamiento eferente y acoplamiento aferente de los paquetes de contenido de método, forman la métrica de: *inestabilidad[57].* Dicha relación significa, que un paquete es estable (un valor cercano a 0) o inestable (un valor más cercano a 1). Como esta ecuación revela, el acoplamiento eferente va *en contra de* la estabilidad de un paquete: Cuanto más un paquete depende de otros paquetes, más susceptible es a los efectos colaterales en los cambios. Por el contrario, si muchos paquetes dependen de un paquete, es menor la probabilidad de futuras modificaciones.
Esta métrica busca calcular el esfuerzo o la facilidad para cambiar un paquete sin afectar a otros paquetes dentro del modelo.

#### 3.3.3.1 Acoplamiento Aferente

En secciones anteriores se había mencionado que el acoplamiento eferente en paquetes se produce cuando dicho paquete hace uso de elementos de otro paquete. En esta sección se aborda que el Acoplamiento Aferente de un paquete de contenido de método se produce cuando otro paquete hace uso de alguno de sus elementos (tareas, roles y artefactos). Por tanto el cálculo del acoplamiento aferente se obtiene mediante la suma de los elementos de otros paquetes que cumplen las características mencionadas anteriormente. En donde se describe el número de dependencias únicas entrantes en un

elemento de método, que hacen alusión al grado de importancia que tienen los elementos en los paquetes del modelo, el cual se obtiene, consultando los elementos predecesores de un elemento i que se encuentran en otro paquete. Por lo tanto, si consideramos que hay n elementos en el paquete, podemos calcular el acoplamiento aferente para un paquete, dependiendo del tipo de elemento, de esta manera:

$$\mathrm{AAe_p} = \sum_{i=0}^{n} \mathbf{Elemento_i}.\,\mathrm{relacionesEntrantesDesdeOtrosPaquetes}, \forall\, Elemento \in p$$

**Fórmula 6. Acoplamiento aferente de elementos en paquetes**

En particular, aplicando la fórmula ( $\mathrm{AAe_p}$) a cada elemento se tiene:

$$\mathrm{AAt_p} = \sum_{i=0}^{n} \mathbf{Tarea_i}.\,\mathrm{relacionesEntrantesDesdeOtrosPaquetes}, \forall\, Tarea \in p$$

**Fórmula 7 . Acoplamiento aferente de tareas en paquetes**

$$\mathrm{AAr_p} = \sum_{i=0}^{n} \mathbf{Rol_i}.\,\mathrm{relacionesEntrantesDesdeOtrosPaquetes}, \forall\, Rol \in p$$

**Fórmula 8 . Acoplamiento aferente de roles en paquetes**

$$\mathrm{AAa_p} = \sum_{i=0}^{n} \mathbf{Artefacto_i}.\,\mathrm{relacionesEntrantesDesdeOtrosPaquetes}, \forall\, Tarea \in p$$

**Fórmula 9. Acoplamiento aferente de artefactos en paquetes**

La inestabilidad de un paquete es la métrica que enfrenta entre sí al acoplamiento y el acoplamiento aferente a través de la siguiente fórmula:

$$\mathrm{I_p} = \frac{\mathrm{Ae_p}}{\mathrm{Ae_p} + \mathrm{AAe_p}}, \forall\, Elemento \in p$$

**Fórmula 10. Inestabilidad de paquetes**

Como la inestabilidad puede tomar cualquier valor entre 0 y 1, en función de a qué extremo se acerque más, podemos darle una interpretación al resultado:

Si la inestabilidad tiende a 0 quiere decir que el paquete tiene un acoplamiento aferente (otros paquetes dependen de él) mucho mayor que el acoplamiento eferente, y por tanto, muchos paquetes de método dependen de él, lo que indica que es altamente responsable, y muy útil para la reutilización en todo el modelo. Estos paquetes gracias a su diseño, por lo general son los más estables del modelo, ya que los paquetes con un

alto nivel de responsabilidad son difíciles de cambiar. Tener demasiada responsabilidad no es necesariamente un mal síntoma. Por ejemplo, paquetes de contenido de método, a menudo están destinados a ser utilizados a lo largo del proceso, y esto incremente su acoplamiento aferente.

En la

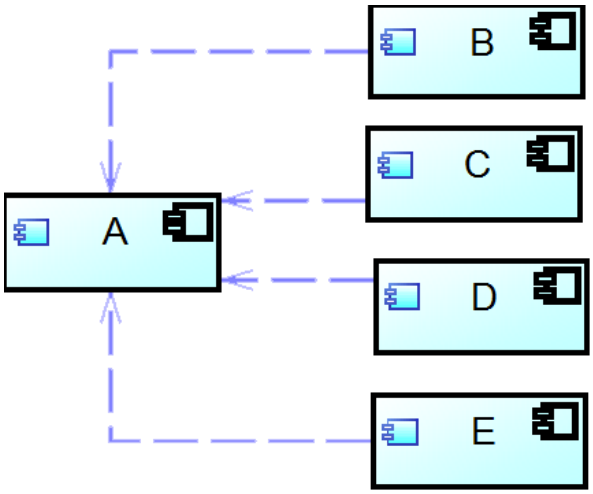



Figura 10, se puede ver que el paquete A, tiene el valor del acoplamiento aferente es 4, y el acoplamiento eferente es 0, siendo un paquete muy estable o con una inestabilidad de 0. El paquete A, es un paquete muy responsable, ya que su comportamiento no depende de otros paquetes, siendo menos propenso al cambio debido a las modificaciones en elementos externos, pero por otro lado, los cambios drásticos en su comportamiento, podría causar un efecto dominó a lo largo de sus cuatro paquetes dependientes, generando inestabilidad a otros paquetes del sistema.

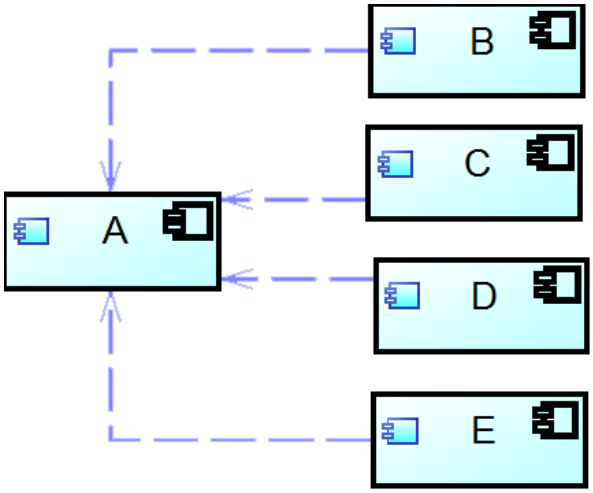
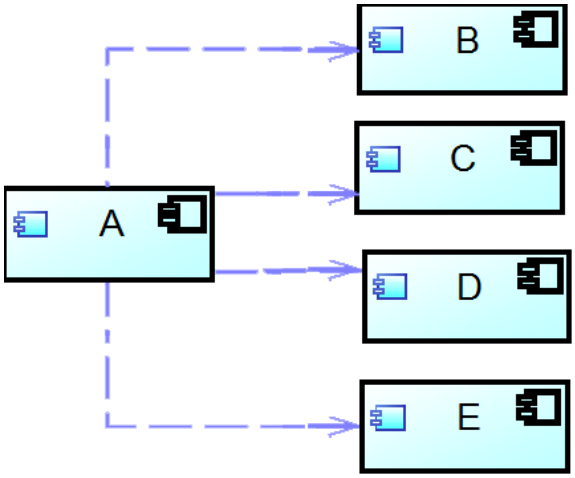



**Figura 10. Paquete estable.**

Por otro lado, si la inestabilidad de un paquete tiende a 1, quiere decir que el acoplamiento eferente es mucho mayor que el aferente o lo que es lo mismo muy pocos o ningún elemento externo al paquete tiene una relación de dependencia hacia un elemento

de él, por lo que el paquete sólo tiene dependencias hacia elementos externos, esto lo convierte en un paquete altamente dependiente de otros en el modelo, los cuales se consideran como inestables, porque modificaciones en otros paquetes del modelo, pueden desencadenar problemas en el comportamiento del paquete en cuestión. Los paquetes inestables, no son lo suficientemente independientes, y además tienen poca responsabilidad con el resto del modelo, otorgándoles más libertad para realizar cambios, ya que no implican un gran impacto en otros paquetes del modelo.




Por ejemplo, la
Figura 11, representa el paquete A, cuyo acoplamiento eferente es 4:

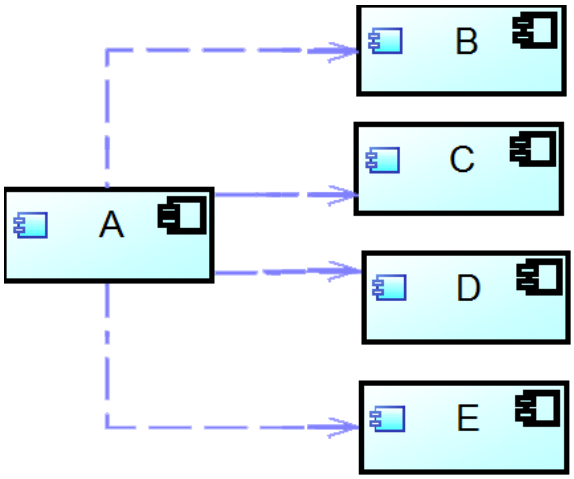
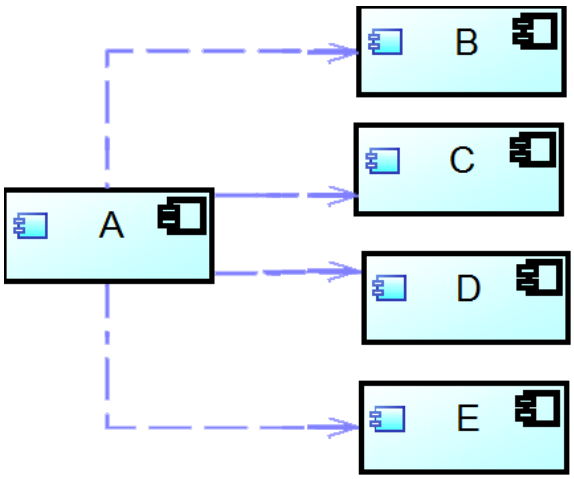



**Figura 11. Paquete Inestable.**




Como muestra la Figura 11, muestra que el paquete A depende de los paquetes B, C, D, E para cumplir con su comportamiento. Como en el caso de acoplamiento aferente, el nivel de dependencia no es algo malo en sí mismo. Es su conocimiento del acoplamiento y cómo los cambios podrían afectar a los otros paquetes, lo que importa.
Y si la inestabilidad tiende a 0.5 por arriba o por abajo, significa que hay un equilibrio entre el acoplamiento eferente y aferente, lo que se podría considerar como el caso ideal.
En resumen, el valor de la inestabilidad cuanto **más** cerca está a cero, más estable y responsable es el paquete, esta indica que hay más dependencias entrantes **(AAe)** que dependencias salientes **(Ae).** Una vez más, paquetes estables son difíciles de cambiar sin afectar el resto de la aplicación. Mientras se acerca a uno, indica más inestabilidad, siendo menos responsables, más dependiente, más fácil de cambiar.

Cuando se diseña un modelo de proceso de software pensando en su modificabilidad, es ventajoso tener paquetes de contenido de método estables porque son menos propensos a cambiar. Del mismo modo, las dependencias de paquetes inestables aumentan el riesgo a efectos colaterales en el modelo en tiempos de cambio.

Nuestra propuesta propone tres maneras de analizar la inestabilidad de los paquetes de contenido de método, de acuerdo a la relación entre elementos del mismo tipo así:

- Inestabilidad del paquete a nivel de tareas
- Inestabilidad del paquete a nivel de roles
- Inestabilidad del paquete a nivel de artefactos

Para poder calcular la inestabilidad de un paquete de contenido de método es necesario calcular el acoplamiento y el acoplamiento aferente de las tareas, roles y artefactos, como lo veremos a continuación

$$\boldsymbol{It_p} = \frac{\mathrm{At_p}}{\mathrm{At_p} + \mathrm{AAt_p}}, \forall\, tarea \in p$$

**Fórmula 11. Inestabilidad de tareas en paquetes**

$$\boldsymbol{Ir_p} = \frac{\mathrm{Ar_p}}{\mathrm{Ar_p} + \mathrm{AAr_p}}, \forall\, rol \in p$$

**Fórmula 12. Inestabilidad de roles en paquetes**

$$\boldsymbol{Ia_p} = \frac{\mathrm{Aa_p}}{\mathrm{Aa_p} + \mathrm{AAa_p}}, \forall\, artefacto \in p$$

**Fórmula 13. Inestabilidad de artefactos en paquetes**

## 3.4 Blueprints de modelos de proceso

Las visualizaciones o Blueprints se realizan a través de un método automatizado que sirve para la evaluación de procesos de software, que usa representaciones gráficas compuestas de nodos y enlaces. Este método ha sido propuesto en este trabajo de grado y ha sido implementado sobre MOOSE, plataforma diseñada para el análisis de datos y software. Los Blueprints de AVIMO-PS utilizan las métricas previamente definidas para dimensionar y graficar los paquetes de contenido de método del modelo bajo análisis.

### 3.4.1 Blueprint de Acoplamiento y Cohesión

El Blueprint de Acoplamiento y Cohesión, tiene tres enfoques, según el tipo de elemento:

- **Roles**: Muestra el proceso enfocado desde los roles que participan en él.
- **Tareas**: Permite visualizar el proceso de desarrollo desde la perspectiva de las tareas que se realizan durante su ejecución.
- **Artefactos**: Muestra el proceso de desarrollo desde el enfoque de los productos de trabajo que se generan en él.

El Blueprint presenta cada paquete de contenido de método como un nodo, cuya dimensión del ancho representan el número de elementos que lo contiene, que pueden ser tareas, roles o artefactos, dependiendo del tipo de visualización, ésta medida da una idea del tamaño del paquete. Por otra parte, la colaboración de dos paquetes, se representa mediante una línea con una punta entre ellos, cuyo grosor varía según el acoplamiento entre los paquetes, para el cálculo del acoplamiento se realiza como se muestra en la sección 3.3.2, ver Figura 12:

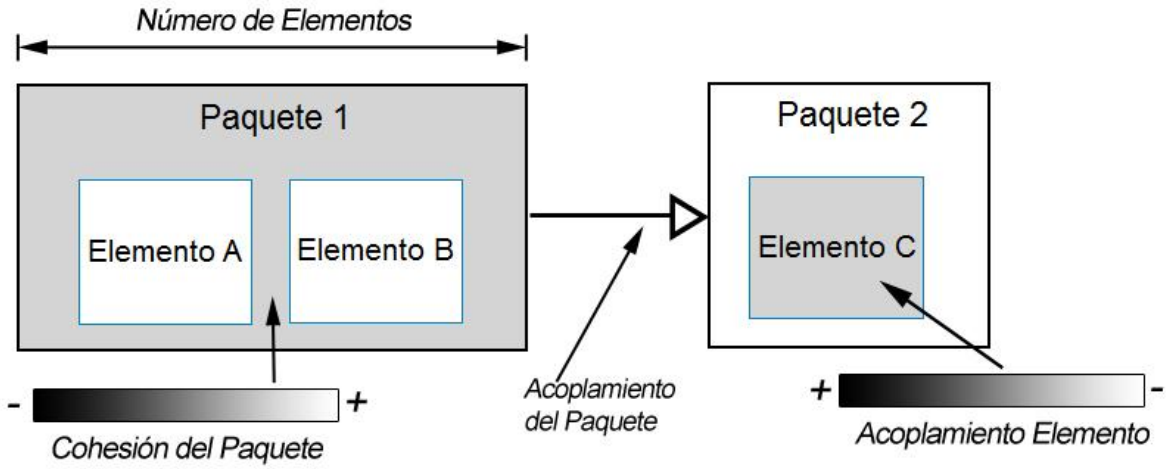



**Figura 12. Representación de nodos y enlaces**

#### 3.4.1.1 Color de los nodos

**Color de los elementos**: Se busca distinguir los elementos que tienen un alto acoplamiento en el paquete, pintándose de color gris, variando la intensidad del nodo, dependiendo del valor de su acoplamiento, así:

$$\mathbf{Elemento}_{\mathrm{i}}.\,\mathrm{relacionesSalientesAOtrosPaquetes}$$

Con el cálculo de la Fórmula 2. Acoplamiento de cualquier elemento en paquetes, se logra obtener los elementos que tienen un alto grado de acoplamiento, o que tienen un alto grado de dependencia en el paquete, ya que dependen considerablemente de otros elementos en otros paquetes.

**Color de los Paquetes**: Se busca distinguir los paquetes que tienen baja cohesión en el modelo, en donde se rellenan de color gris, variando la intensidad del nodo, dependiendo de la cohesión.

Para ello se utiliza la métrica de Cohesión Fórmula 1, para poder resaltar los nodos de los paquetes en un esquema de colores en escala de grises, la intensidad del color gris de cada nodo es inversamente proporcional a la cohesión, es decir a mayor cohesión menor intensidad de color gris (tendencia al color blanco), y a menor cohesión más oscuro será el nodo (tendencia al color negro), así:

**Color Blanco**: El paquete es de color blanco, si ninguno de sus elementos tiene relaciones que salen hacia otros paquetes. Lo que es lo mismo

$$\boldsymbol{Cohesion_{P_i} = 1}$$

**Color Gris:** El color gris depende del número de elementos con la mayoría de relaciones hacia elementos de otros paquetes. La intensidad del color gris está dada por*:*

$$\textbf{Intensidad} = 1 - \textbf{Cohesion}_{\textbf{Pi}}$$

### 3.4.1.2 Enlaces entre nodos (Acoplamiento)

El acoplamiento se puede observar gráficamente mediante una línea con punta que une los paquetes acoplados. El grosor de la línea de dependencia entre los paquetes varía según el número de relaciones salientes a elementos en otros paquetes. La métrica de Acoplamiento entre Paquetes de Contenido de Método propuesta en la sección 3.3.2, mide el número de elementos de método, de los cuales depende un elemento cualquiera, entonces el grosor del enlace dependerá de la Fórmula 2 ($\mathbf{Ae_p}$)

$$\mathrm{Ae_p} = \sum_{i=0}^{n} \mathbf{Elemento_i}.\,\mathrm{relacionesSalientesAOtrosPaquetes}\,, \forall\, Elemento \in p$$

### 3.4.1.3 Visualización en AVIMO

El Blueprint de Acoplamiento y Cohesión se puede generar, dependiendo del tipo de elemento: Rol, Tarea y Artefacto. La Figura 13. Blueprint de acoplamiento y cohesion en paquetes - tareases un Blueprint de acoplamiento y cohesión, del modelo de proceso de software OpenUp[16], enfatizando en las tareas.

[16] http://www.eclipse.org/epf/downloads/praclib/praclib_downloads.php

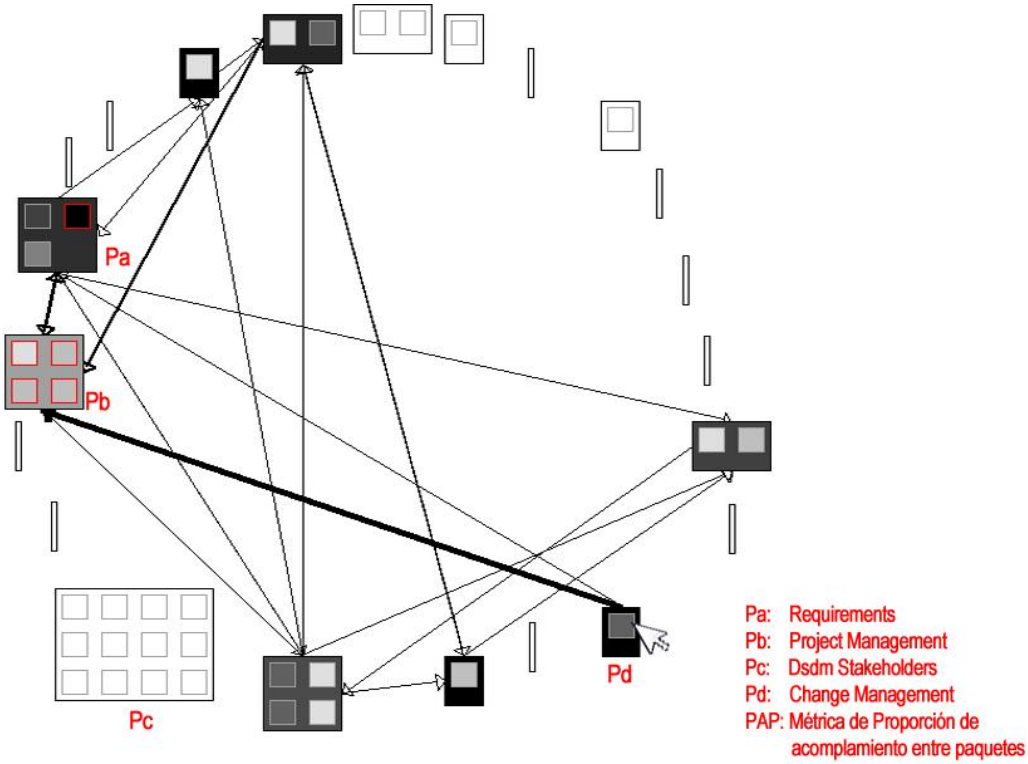



**Figura 13. Blueprint de acoplamiento y cohesion en paquetes - tareas**

**Descripción:** El Blueprint de Acoplamiento y Cohesión, tiene como objetivo ayudar en el análisis visual de la modularidad del modelo de proceso de software, en donde los rectángulos contenedores, representan los paquetes de contenido de método, cuyo ancho está dado por la cantidad de elementos en el paquete. Entre más oscuro sea el color del paquete, refleja una menor cohesión del mismo, o lo que es igual, que sus elementos tienen la mayoría de relaciones con elementos de otros paquetes.

Una línea que une dos paquetes, implica que elementos de un paquete consume elementos del otro, lo que conlleva a un acoplamiento entre dichos paquetes. Entre mayor sea el acoplamiento más gruesa será la línea que los une, la flecha de la línea apunta hacia el paquete servidor (paquete que contiene los elementos consumidos). Por otro lado, los nodos contenidos dentro de los paquetes, son elementos de contenido de método y se representan con un cuadro cuyo color de relleno está dado por el acoplamiento a elementos en otros paquetes, a mayor acoplamiento, mayor intensidad del color gris, siendo de gran utilidad para poder identificar los elementos que están aumentando el acoplamiento en los paquetes.

Para una mayor simplicidad y facilitar la lectura y comprensión del Blueprint, no se grafican las relaciones entre los elementos de contenido de método (roles, tareas, artefactos),

**Ejemplo:** La Figura 13, muestra el Blueprint de acoplamiento y cohesión generado para el modelo de proceso de software OpenUp, haciendo énfasis solo en las tareas del modelo.

En el ejemplo se observa que el paquete Pd (Change Management) es de color negro, indicando que tiene una cohesión muy baja, a diferencia del paquete Pc (Dsdm Stakeholders) que es de color blanco presentando una cohesión muy alta (Cohesión ideal) ya que sus tareas se relacionan principalmente solo con tareas del mismo paquete.
Para poder ver los elementos a los que está acoplado un elemento cualquiera, solo basta posicionar el cursor sobre un elemento, en donde automáticamente se resaltan en color rojo los elementos de los cuales depende. En el ejemplo el cursor se encuentra sobre la tarea del paquete Pd, mostrando que dicha tarea depende de las tareas resaltadas en rojo de los paquetes Pa y Pb.

**Interpretación**: Estos Blueprint ayudan a evaluar si la modularidad de los modelos de procesos de software es apropiada para facilitar la mantenibilidad. Además permiten identificar paquetes y/o elementos altamente acoplados, o con muy baja cohesión, para que estos se sometan a un análisis especial por parte del ingeniero de procesos para encontrar oportunidades de mejora. Los elementos de contenido de método que estén altamente acoplados a un paquete en particular se pueden mover hacia el otro paquete para disminuir de esta forma el acoplamiento e incrementar la cohesión.

## 3.5 Blueprint Inestabilidad

Este Blueprint muestra la inestabilidad de cada paquete de contenido de método, basándose para ello en la Fórmula 10. Inestabilidad de paquetes, propuesta en este trabajo, ver sección 3.3.3. El Blueprint presenta cada paquete de contenido de método como un nodo, cuya dimensión del ancho representan el número de elementos que lo contiene, que pueden ser tareas, roles o artefactos, dependiendo del tipo de visualización. Además, la colaboración de dos paquetes, se representa mediante una línea con una punta entre ellos, cuyo grosor varía según el acoplamiento entre los paquetes, para el cálculo del acoplamiento se realiza utilizando la Fórmula 2, como se muestra en la sección 3.3.2 (ver Figura 14. Representación de nodos y enlaces):

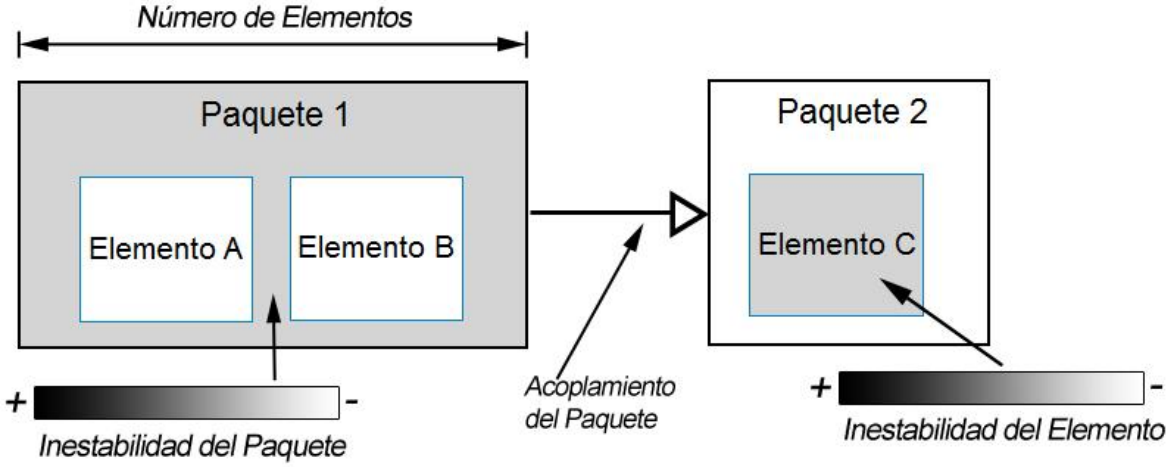



**Figura 14. Representación de nodos y enlaces**

Después de dejar por sentados los conceptos de acoplamiento eferente y acoplamiento aferente aplicados a tareas, roles y artefactos, podemos entrar a analizar nuestro conjunto de medidas que nos aporten la información necesaria para poder detectar posibles anomalías en los paquetes de contenido de método de los procesos. Es importante recordar que la inestabilidad es una medida que calcula un ratio entre el Acoplamiento eferente (que lleva hacia afuera) con el Acoplamiento aferente (que llega)

### 3.5.1 Color de los nodos

**Color de los elementos**: Se busca distinguir los elementos inestables, o que cuyo comportamiento dependa de elementos externos, y por ende puede ser víctima de efectos colaterales, estos nodos se pintan de color gris, variando la intensidad del nodo, dependiendo del valor de su inestabilidad, así:

$$\mathbf{Elemento}_{\mathrm{i}}.\,\mathrm{inestabilidad}$$

La métrica de inestabilidad puede destacar un elemento y ver que tan inestable es dentro del proceso. Es decir un elemento con una alta inestabilidad, su funcionamiento estará guiado por el comportamiento de terceros, por lo tanto, el mal funcionamiento de elementos externos puede causar un mal funcionamiento en su comportamiento, lo que a su vez puede poner en peligro el funcionamiento global del proceso. Tales elementos deben de tener una atención especial.

**Color de los paquetes**: Para visualizar la métrica de inestabilidad de los paquetes, utilizaremos el color de los nodos:

- Si el paquete es estable se usa el color blanco (valor 0).
- Si es inestable el negro (valor 1).
- Si toma un valor intermedio se mostrará de color gris.

La intensidad de los nodos de los paquetes está dada por la Fórmula 10. Inestabilidad de paquetes, para poder resaltar los nodos en un esquema de colores en escala de grises, así:

**Color Blanco**: Significa que uno o varios nodos (paquete o elemento) dependen de él, pero él no depende de nadie. Es responsable e independiente, convirtiéndose en un nodo completamente estable. Lo que es lo mismo, una inestabilidad Cero o Nula $\mathbf{I_P = 0}$

**Color Gris**: El color gris se intensifica según el grado de inestabilidad, que ningún o pocos nodos dependen del nodo que se está midiendo, y por el contrario este nodo si depende de muchos otros, convirtiéndose en inestable, es decir, es dependiente y no responsable. La intensidad está dada por la Fórmula 11:

$$\boldsymbol{It_p} = \frac{\mathrm{At_p}}{\mathrm{At_p} + \mathrm{AAt_p}}, \forall\, tarea \in p$$

### 3.5.2 Enlaces entre nodos (Acoplamiento Aferente)

Las relaciones entre los paquetes, al igual que en el Blueprint de Acoplamiento y Cohesión, representa el acoplamiento que existe entre los paquetes de contenido de método. El grosor de la línea de dependencia entre los paquetes varía según el número de relaciones salientes a elementos en otros paquetes. La métrica de Acoplamiento entre Paquetes de Contenido de Método propuesta en la sección 3.3.2, mide el número de elementos de método, de los cuales depende un elemento cualquiera, entonces el grosor del enlace dependerá de la Fórmula 2.

$$\mathrm{Ae_p} = \sum_{i=0}^{n} \mathbf{Elemento_i}.\,\mathrm{relacionesSalientesAOtrosPaquetes}, \forall\, Elemento \in p$$

**Fórmula 2. Acoplamiento de cualquier elemento en paquetes**

### 3.5.3 Visualización en AVIMO-PS

El Blueprint de Inestabilidad se puede generar, dependiendo del tipo de elemento: Rol, Tarea y Artefacto. En la Figura 15 se presenta un Blueprint de inestabilidad del modelo de proceso de software OpenUp, haciendo énfasis en las tareas.

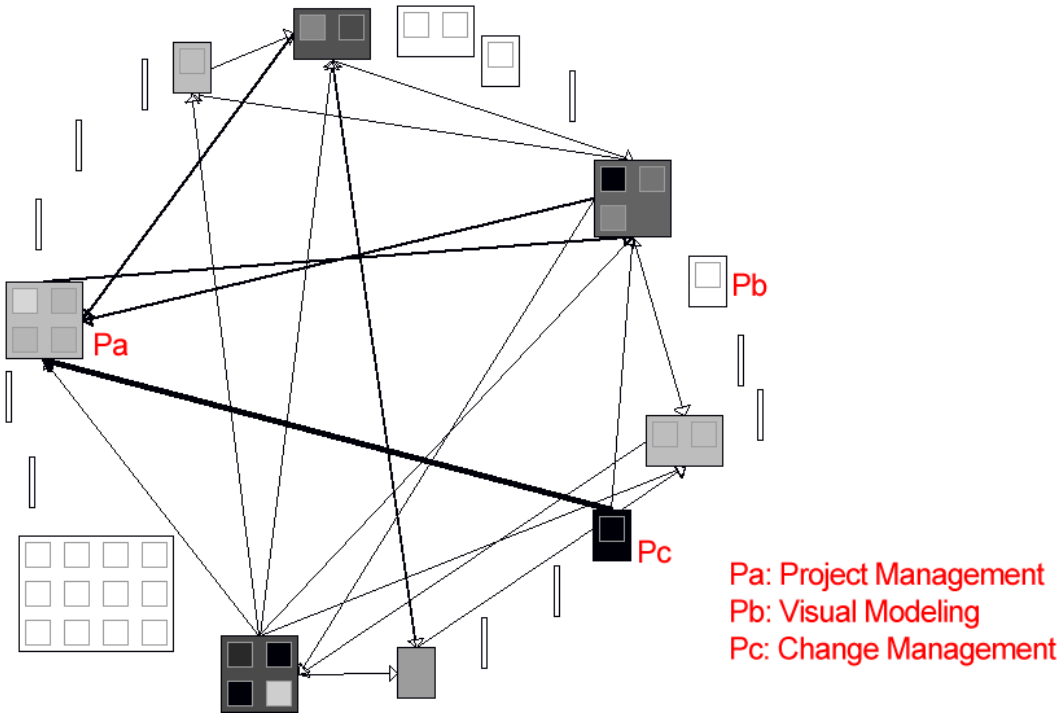



**Figura 15. Blueprint de inestabilidad aplicado a OpenUp**

**Descripción:** El Blueprint de Inestabilidad, tiene como objetivo ayudar a identificar los paquetes y/o elementos de contenido de método que son inestables o no responsables en un modelo de proceso de software, en donde los rectángulos contenedores, representan los paquetes de contenido de método, cuyo ancho está dado por la cantidad de elementos en el paquete. Entre más oscuro sea el color del paquete, refleja una mayor inestabilidad, o lo que es lo mismo, que sus elementos tienen un alto grado de dependencia hacia elementos de otros paquetes, con relación a los paquetes que dependen de él.

Una línea que une dos paquetes, implica que elementos de un paquete consume elementos del otro, lo que conlleva a un acoplamiento entre dichos paquetes. Entre mayor sea el acoplamiento más gruesa será la línea que los une, la flecha de la línea apunta hacia el paquete servidor (paquete que contiene los elementos consumidos).

Por otro lado, los nodos contenidos dentro de los paquetes son elementos de contenido de método y se representan con un cuadro, cuyo color de relleno está dado por el grado de inestabilidad, a mayor inestabilidad mayor intensidad del color gris, siendo de gran utilidad para poder identificar los elementos que conllevan a los paquetes hacia la inestabilidad. Para una mayor simplicidad y facilitar la lectura y comprensión del Blueprint, no se grafican las relaciones entre los elementos de contenido de método (roles, tareas, artefactos),

**Ejemplo:** La Figura 15 muestra un Blueprint de Inestabilidad, el cual fue generado para el modelo de proceso de software OpenUp haciendo énfasis solo en las tareas del modelo. En el ejemplo se observa que el paquete Pc (Change Management), es de color negro, indicando que tiene una inestabilidad muy alta, indicando que sus elementos de contenido de método son muy inestables y no responsables, a diferencia del paquete Pb “Visual Modeling” que es de color blanco presentando una inestabilidad muy baja, ya que sus tareas, no dependen de tareas de paquetes externos, convirtiendo al paquete en un paquete responsable y con menos probabilidad al cambio ocasionada por paquetes externos. Para poder ver los elementos a los que está acoplado un elemento cualquiera, solo basta posicionar el cursor sobre un elemento, automáticamente se resaltan en color rojo los elementos de los cuales se relaciona.

**Interpretación**: la métrica de inestabilidad facilita la localización de paquetes y/o elementos de contenido de método oscuros a los que se deben prestar mucha atención, ya que son inestables, con el fin de que estos sean valorados y analizados en contexto por un ingeniero de procesos y así poder encontrar oportunidades de mejora.

## 3.6 Prototipo AVIMO-PS

Durante la exploración de las herramientas que permiten la importación y posterior visualización de modelos de procesos software, seleccionamos AVISPA, dado que es una

herramienta con un entorno ágil y ligero que permite recuperar, analizar y construir Blueprints, a partir de la especificación SPEM2.0 de los modelos de proceso. AVISPA está implementada sobre la plataforma de código abierto Moose, por ello se realiza una extensión que permite agregar nuevas funcionalidades para un análisis visual de la modularidad de los paquetes de contenido de método en los modelos de procesos.
El primer requerimiento surgió al momento de realizar la importación de la especificación del proceso XML, ahora considerando nuevos elementos del proceso SPEM2.0. Implementar este requerimiento requirió un gran esfuerzo de desarrollo puesto que tardamos mucho en superar la curva de aprendizaje del lenguaje Smalltalk, ya que no existe suficiente documentación sobre la plataforma. Para ello fue necesario dedicar ciertas sesiones de desarrollo con el director del proyecto quién había participado en la formulación e implementación de AVISPA.

Después de extender la importación del modelo de proceso, se procede a resolver las referencias de todos los elementos de contenido de método, haciendo hincapié en los paquetes de contenido de método, dado que nuestra propuesta analiza dichos paquetes, y es sobre estos que se aplican las métricas propuestas en la Sección 3.3. Previamente debió extenderse el Meta Modelo de AVISPA para soportar la organización por paquetes, para ello se diseñaron e implementaron nuevas clases y se modificaron otras clases que son parte del core de AVISPA. Las visualizaciones fueron implementadas en el motor de visualización de Mondrian.

En esta sección se enfatiza en el funcionamiento interno del prototipo AVIMO-PS, dando una breve descripción de los distintos componentes que fueron creados y extendidos sobre la implementación previa de AVISPA.

### 3.6.1 Diseño e implementación de AVIMO-PS

AVIMO-PS es una extensión de la herramienta AVISPA, por lo tanto hereda sus características básicas y arquitectónicas, combinando la meta-modelización y técnicas de visualización.

Este trabajo de grado propone un conjunto de métricas para medir la modularidad de los paquetes de contenido de método de cualquier modelo de proceso de software definido en SPEM2.0. Dicho modelo está descrito en un archivo XML, el cual se analiza y cada elemento estructural se materializa para formar una instancia del meta-modelo SPEM 2.0, dando el resultado de las mediciones en una representación gráfica o Blueprint. AVIMO-PS se construye sobre de la plataforma Moose, que es el que permite la importación, análisis de datos, modelado, medición, consulta, y Mondrian que trabajando sobre Moose facilita la construcción de herramientas de análisis visual e interactiva.

Se espera con este trabajo de grado que el prototipo AVIMO-PS se convierta en una herramienta fiable para importar y analizar la modularidad de paquetes de contenido de método en modelos de procesos de software especificados en el estándar SPEM 2.0. Su

panel de navegación muestra cuatro puntos de entrada (actividades, artefactos, paquetes de contenido de método, roles y tareas) para iniciar el análisis, el cual se realiza a través de la información disponible en el Meta Modelo, siendo los paquetes de contenido de método la funcionalidad principal que AVIMO-PS ha extendido de AVISPA.

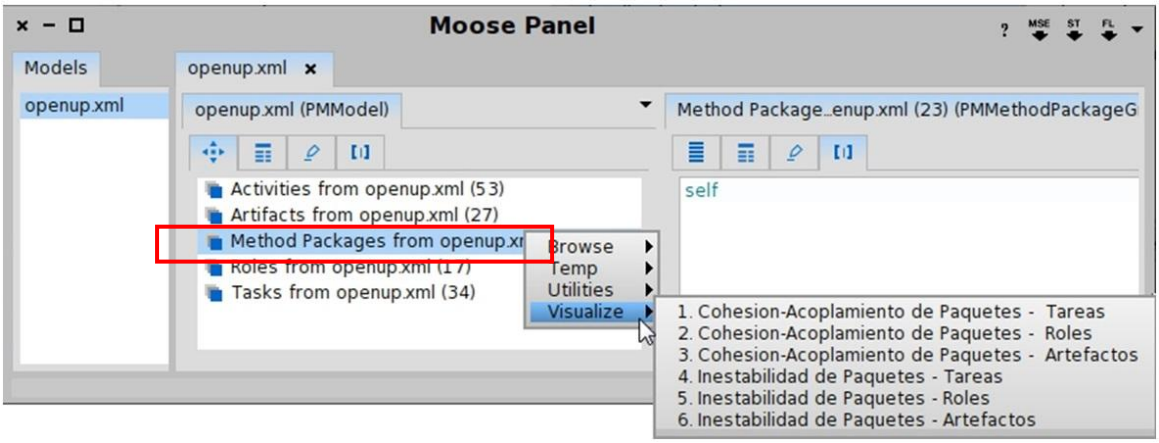



**Figura 16. Interfaz principal del usuario en AVIMO**

A nivel de arquitectura, AVIMO-PS mantiene la misma distribución de los módulos que tiene la herramienta AVISPA, siguiendo un patrón en capas y de generalización, como se muestra en la Figura 17. Cada módulo de AVIMO-PS se implementa particularizando elementos definidos en módulos abstractos de Moose. Desde una perspectiva de componentes y conectores el patrón que sigue la herramienta es de tuberías y filtros.

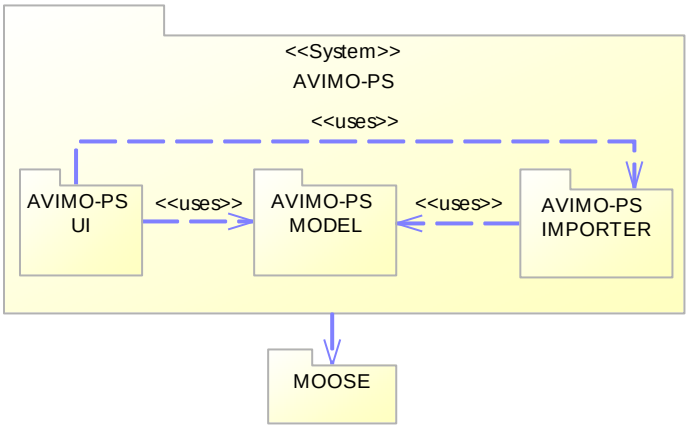



**Figura 17. Vista de los módulos de AVIMO-PS**

El módulo *AVIMO-PS Importer* tiene la responsabilidad de crear o materializar los objetos necesarios, tomando como punto de partida un modelo de procesos de software, proporcionado por Eclipse Process Framework Composer (EPFC), dicho modelo debe estar en formato XML.

El módulo *AVIMO-PS Model* tiene la responsabilidad de representar los modelos de proceso y calcular las métricas de modularidad a los objetos que representan los paquetes de contenido de método y sus elementos relacionados.

*AVIMO-PS UI* incluye secuencias de comandos de navegación y visualización. Estas funcionalidades se ilustran en la Figura 18.

Los importadores, modelos, navegación y visualización implementados en la herramienta proporcionan una gran ayuda al ingeniero de procesos, para poder analizar de una manera visual la modularidad de los modelos de proceso.

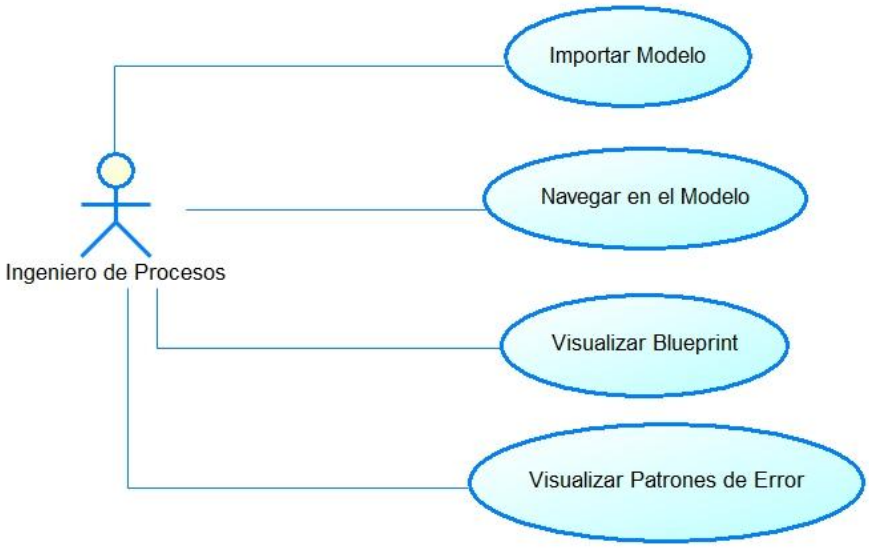



**Figura 18. Diagrama de casos de uso de AVIMO-PS**

La Figura 18, presenta la funcionalidad del prototipo AVIMO-PS, la cual está dirigida a ingenieros de procesos, el cual tiene la función de modelar el proceso de software de la organización y establecer guías de adaptación del proceso a proyectos específicos. El ingeniero de procesos puede crear, modificar y eliminar elementos de un modelo de proceso software basándose en el lenguaje de metamodelado SPEM2.0.

A continuación se menciona el proceso que se lleva a cabo para importar un modelo de proceso de software desde su especificación en XML:

Todo inicia cuando el ingeniero de procesos ha definido el proceso en la herramienta EPFC, éste es ingresado a la herramienta AVIMO-PS a través del caso de uso Importar modelo, una vez importado el modelo de proceso (ver Figura 19), se procede a leer la información del archivo, el cual contiene la especificación SPEM 2.0 del modelo del proceso, y se realiza el tratamiento de importar todos sus elementos y crear los objetos en la herramienta. Cada elemento del proceso se importa y almacena en colecciones que se usan para construir el modelo de proceso. Después de ello, las relaciones internas se resuelven y las métricas se inicializan. El ingeniero de procesos, puede navegar entre los diferentes elementos de proceso soportados por la herramienta: Tareas, Roles, Artefactos, Paquetes de contenido de método, con el fin de conocer más a fondo sus características y relaciones, a través del caso de uso Navegar en el modelo.

Finalmente, el Blueprint del modelo instalado en la plataforma Moose puede ser desplegado. Dichos Blueprint pueden contener información de la modularidad del modelo del proceso y/o patrones de error, a los cuales el ingeniero de proceso puede acceder gracias a los casos de uso Visualizar Blueprint y Visualizar patrones de error (Ver Figura 15).

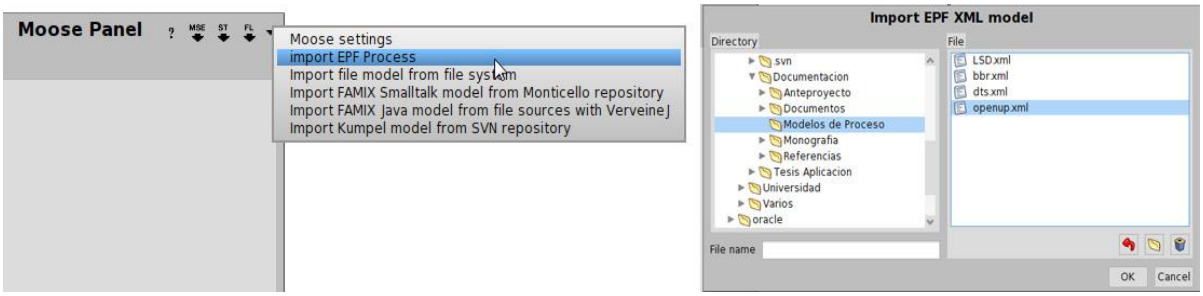



**Figura 19. Importar un modelo de proceso de software a AVIMO-PS**

AVIMO-PS ha extendido el metamodelo de AVISPA, como se muestra en la Figura 20, para poder analizar y representar los paquetes de contenido de método. AVISPA se extiende por medio de subclases de MooseEntity y MooseGroup. La nomenclatura que se utiliza para el nombrado, tanto como en AVISPA como en AVIMO-PS, es el prefijo PM en el nombre de las clases.

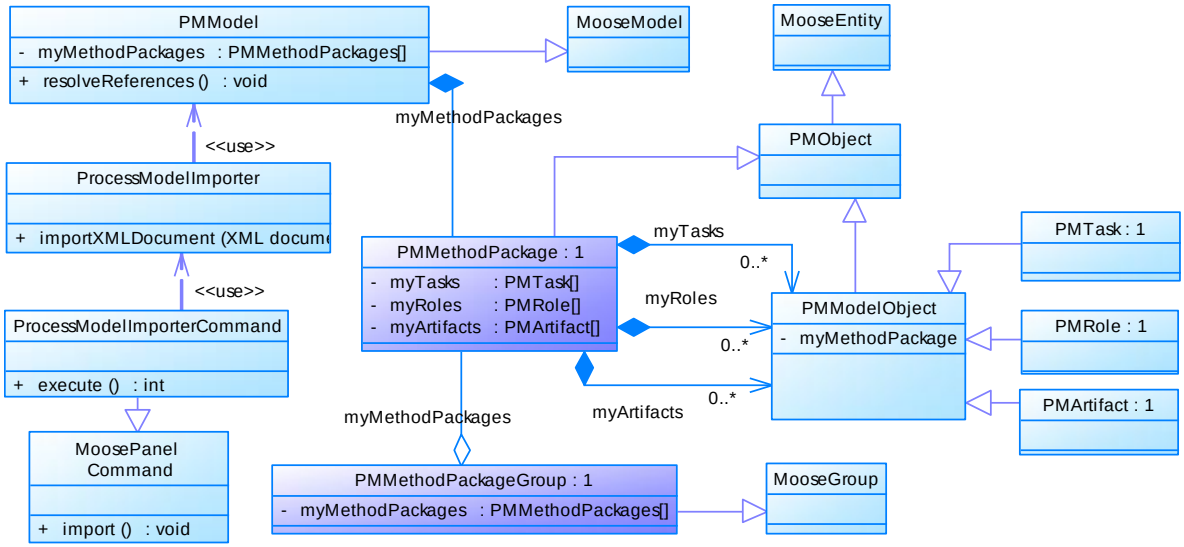



**Figura 20. Diagrama parcial de clases de AVIMO-PS**

La clase PMObject contiene todas las operaciones y atributos comunes a todos los elementos de contenido de método de SPEM2.0 (esencialmente un identificador, nombre, descripción, identificador del paquete al que pertenecen). Por otro lado las clases PMRole, PMTask PMArtifact, describen los elementos de contenido de método, y junto con la clase PMMethodPackage, detallan sus relaciones, en donde cada una de estas clases tiene los atributos y métodos para el cálculo de métricas, visualización y navegación a través del modelo. El grupo de paquetes de contenido de método se

expresan como instancias de la clase PMMethodPackageGroup, con el propósito de crear colecciones especializadas y permitir la visualización de los Blueprint, a través de sus métodos.

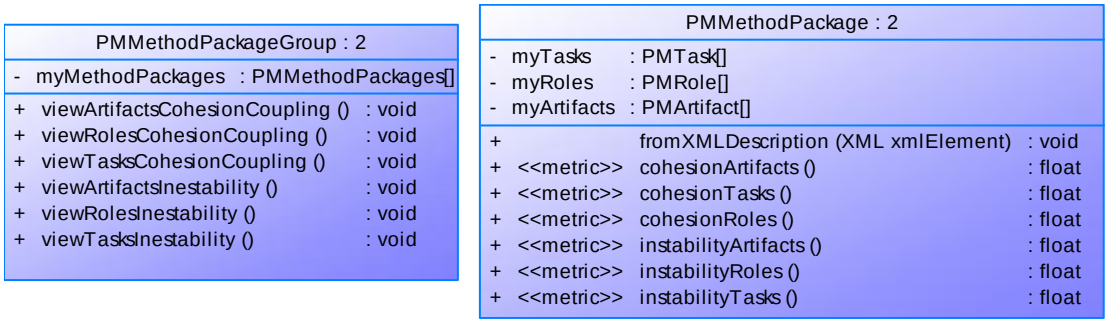



**Figura 21. Detalle de las clases principales de AVIMO-PS**

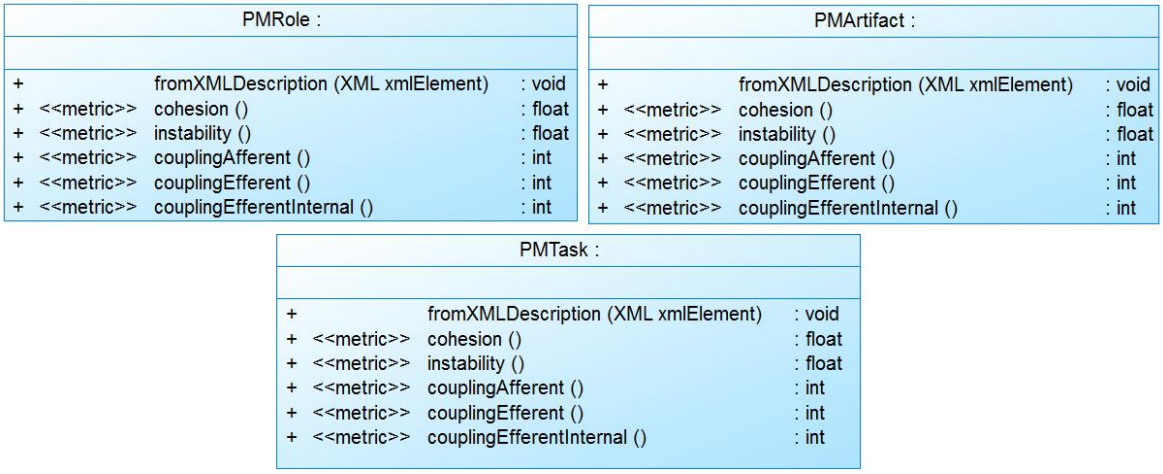



**Figura 22. Extensiones a las clases principales de AVISPA**

### 3.6.2 Realización de los casos de uso a nivel de diseño e implementación

La Figura 23 muestra la secuencia de los mensajes, cuando un modelo de proceso de software es importado a AVIMO-PS. Todo inicia cuando el actor selecciona el modelo del proceso a importar, el cual es capturado por un objeto ProcessModelImporterCommand, el cual delega la petición de importar a un objeto ProcessModelImporter, a través del método ImportXMLDocument. Dicho método, recorre los paquetes de contenido de método que se encuentran en el documento XML (especificación del modelo de proceso), y lo materializa mediante el método fromXMLDescription, este método recibe como parámetro un objeto XML que contiene la especificación del paquete de contenido de método, el cual se somete a un análisis, en busca de información que es relevante para la clase PMMethodPackage, como lo son los elementos Tareas (Task), Roles (Role) y Artefactos (Artifact), y por cada uno de estos elementos encontrados se materializan invocando el método fromXMLDescription de cada clase (PMTask, PMRole, PMArtifact), almacenándose en colecciones en el objeto PMMethodPackage, que se usan para construir el modelo de proceso.

Una vez materializados los paquetes de contenido de método y sus elementos, las relaciones internas del modelo de proceso se resuelven utilizando el método resolveReferences y se calculan las métricas necesarias para la construcción de los Blueprint. Finalmente, el modelo se instala en la plataforma Moose y se añade al MoosePanel mediante el objeto ProcessModelImporterCommand.

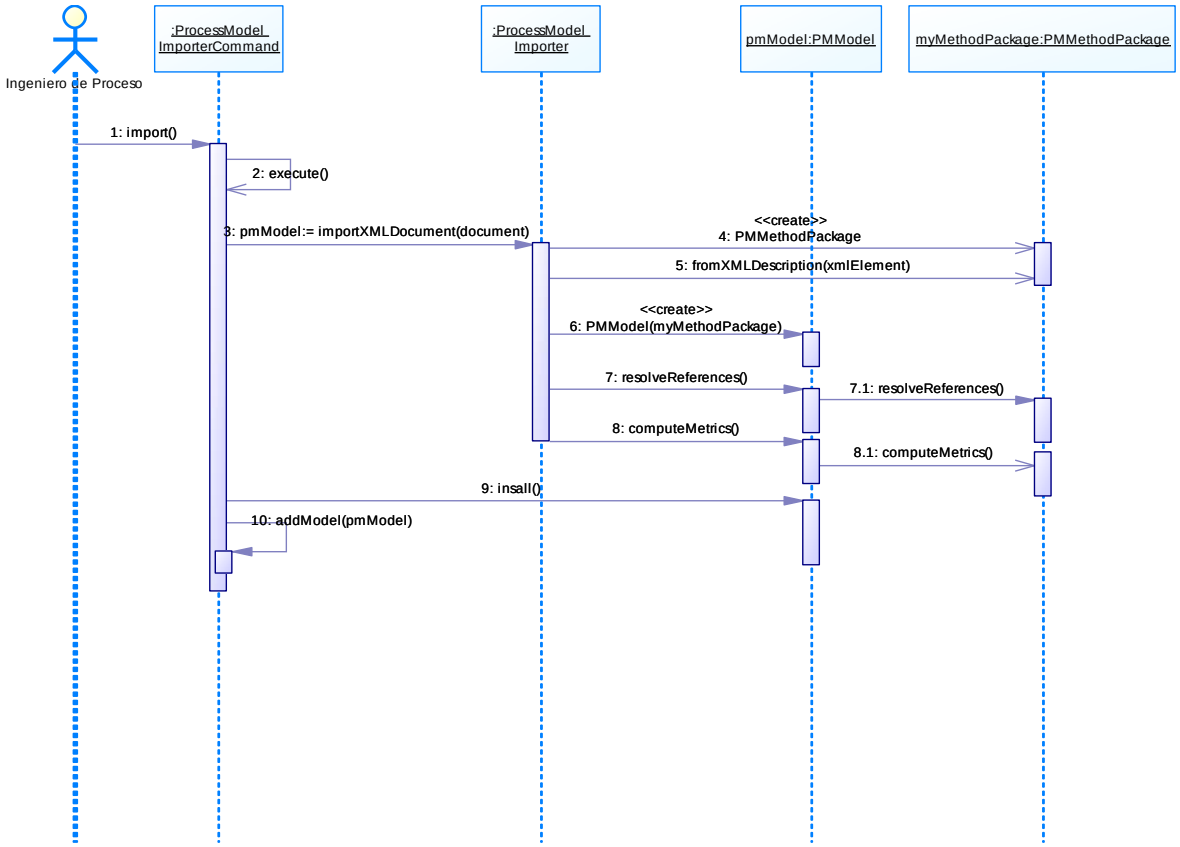



**Figura 23. Importando un proceso EPF en AVIMO-PS**

El mensaje 3 de la Figura 23: *importXMLDocumet* de la clase ProcessModelImporter, es un servicio que lee la información de un documento XML, documento que ya ha sido seleccionado por el usuario, a través de los servicios que proporciona la clase ProcessModelImporterCommand. *ImportXMLDocumet* obtiene los elementos principales para el análisis: `ContentElement, MethodPackage`, con los cuales se procesan y materializan para tener una representación del modelo en XML en objetos en AVIMO-PS. Primero se obtienen una lista de los elementos XML que sean del tipo `ContentElement y MethodPackage`, después de ello, se extraen todas las tareas (Task), roles (Role) y artefactos (Artifact) de la lista `contentElements`. Del mismo modo, se extraen todos los paquetes de contenido de método de la lista `methodPackage`. Finalmente se crea un objeto ***PMModel*** asignándole todos los elementos encontrados en el modelo XML. A continuación se muestra un fragmento de código donde se observan las instrucciones más importantes al momento de importar un modelo EPFC.

**ProcessModelImporter>> importXMLDocument (XML document)**

```
1.  contentElements := document // #ContentElement.
2.  methodPackage := document // #MethodPackage.

3.  tasks  := contentElements select:
4.          [:el | ((el attributeAt: #xsi:type)  = 'uma:Task' ) ].
5.  pmTasks := tasks collect:
6.          [:xmlElement | PMTask fromXMLDescription: xmlElement].
7.  roles := contentElements select:
8.          [:el | ((el attributeAt: #xsi:type) = 'uma:Role' ) ].
9.  pmRoles := roles collect:
10.         [:xmlElement | PMRole fromXMLDescription: xmlElement].
11. artifacts := contentElements select:
12.         [:el | ((el attributeAt: #xsi:type) ='uma:Artifact' ) ].
13. pmArtifacts := artifacts collect:
14.         [:xmlElement | PMArtifact fromXMLDescription: xmlElement].

15. methodPackages  := methodPackage select:
16.         [:el | (el attributeAt: #xsi:type) = 'uma:ContentPackage' ].
17. pmMethodPackage := methodPackages collect:
18.         [:xmlElement | PMMethodPackage
19.         fromXMLDescription: xmlElement].

20. pmModel := PMModel new
21.              tasks: pmTasks;
22.              roles: pmRoles;
23.              artifacts: pmArtifacts;
24.              methodPackages: pmMethodPackage.
```

En el fragmento de código anterior, se visualiza en la línea 19, que al obtener los paquetes de contenido de método, se hace un llamado al método `fromXMLDescription` de la clase `PMMethodPackage`. Esto es necesario, para poder leer toda la información de un paquete, el cual contiene tareas, roles, artefactos, descripción, etc.
A continuación se esboza cómo los paquetes obtienen la información necesaria para calcular las métricas y con ello generar los Blueprint. Es aquí en donde se agregan los elementos de contenido de método a los paquetes respectivos.

```
1.  PMMethodPackage: fromXMLDescription( XMLElement xmlElement)
2.  answer := self new.
3.  answer mooseName: (xmlElement attributeAt: #name).
4.  answer id: (xmlElement attributeAt: #id).
5.  answer presentationName: (xmlElement attributeAt: #presentationName).


6.  tasksPastell  := ( (xmlElement / #ContentElement) select: [:el | (el
7.  attributeAt: #xsi:type) = 'uma:Task' ] )
8.    collect: [:item | PMTask  fromXMLDescription: item ].
```

```
9.  rolesPastell  := ( (xmlElement / #ContentElement) select: [:el | (el
10. attributeAt: #xsi:type) = 'uma:Role' ] )
11.   collect: [:item | PMRole  fromXMLDescription: item ].

12. artifactsPastell  := ( (xmlElement / #ContentElement) select: [:el |
13. (el attributeAt: #xsi:type)= 'uma:Artifact'  ] )
14.   collect: [:item | PMArtifact  fromXMLDescription: item ].

15. answer tasks: tasksPastell.
16. answer roles: rolesPastell.
17. answer artifacts: artifactsPastell.
```

La

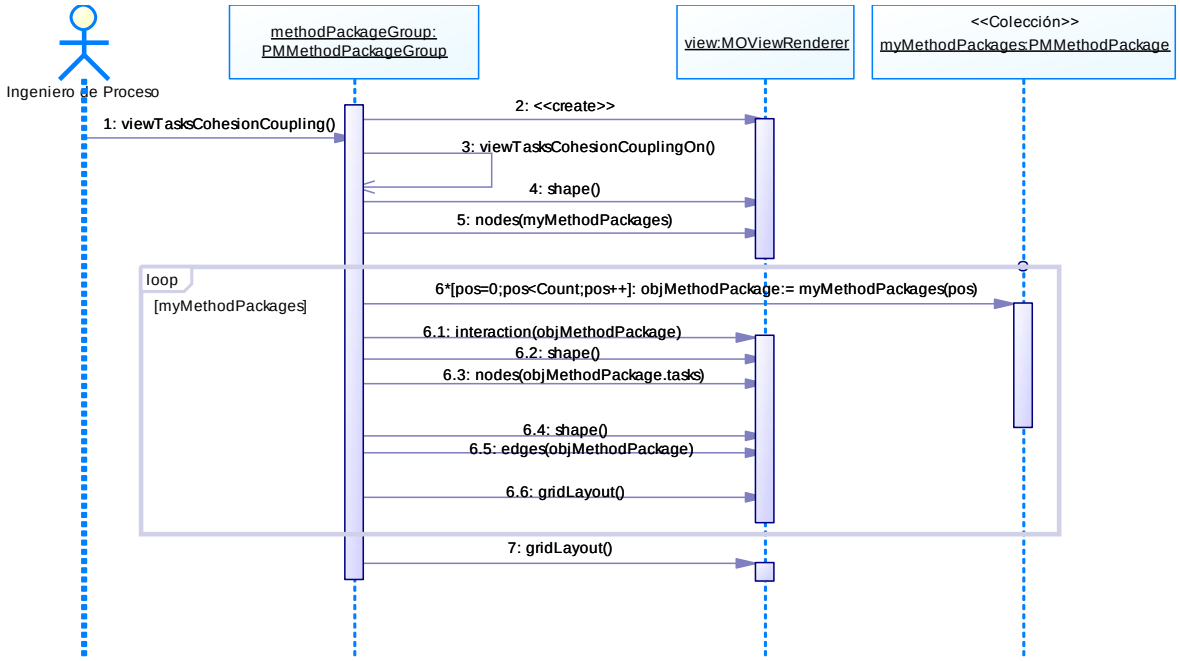



Figura 24, muestra cómo se realiza una visualización del Blueprint de acoplamiento y cohesión de tareas. La secuencia se inicia cuando el usuario, situado en *PMMethodPackageGroup* selecciona la opción de visualizar y selecciona *viewTaskCohesionCoplingBlueprint* (Blueprint de acoplamiento y cohesión de tareas). El objeto *PMMethodPackageGroup* crea una vista (tipo de *MOViewRenderer*) y ejecuta o activa el método *viewTaskCohesionCoplingBlueprintOn*. Este método utiliza la vista creada previamente para generar el Blueprint y ser desplegado al usuario. En las figuras Figura 20 y Figura 21Figura 20,  se resumen las clases en un diagrama de clases UML y se muestra el comportamiento de las distintas clases de acuerdo con los escenarios descritos. También define las relaciones entre las clases principales AVIMO-PS.

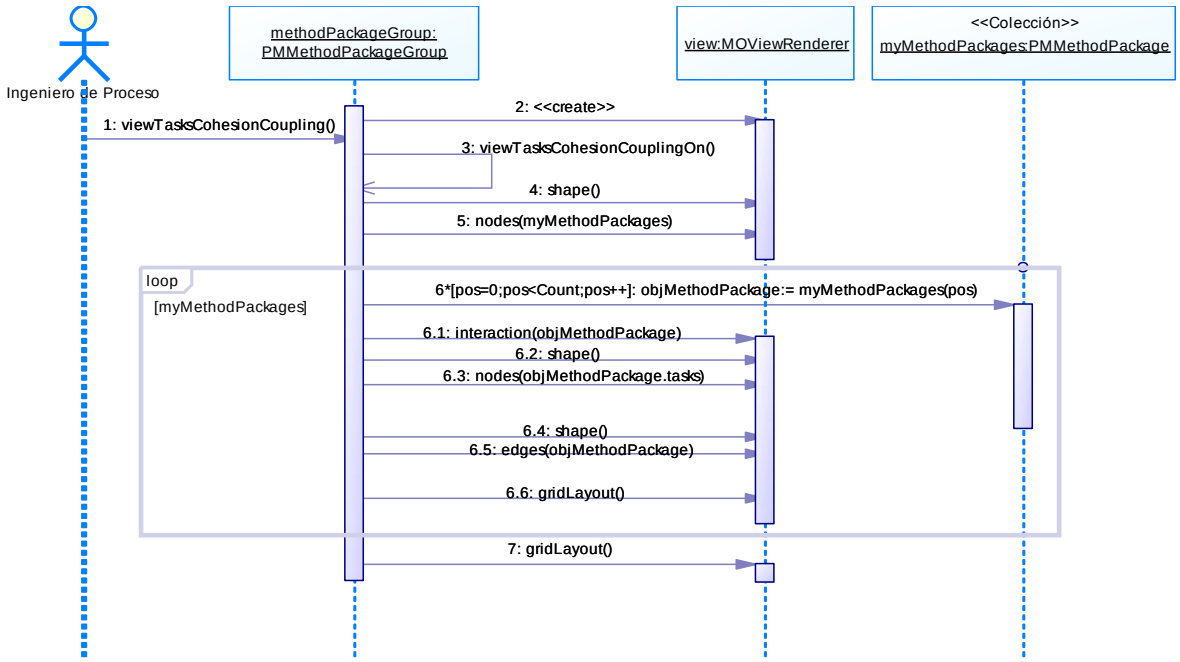



**Figura 24. Visualización del Blueprint de Acoplamiento y Cohesión de tareas en AVIMO-PS**

# 4 Estudio de Caso: Análisis de la modularidad de modelos de proceso de software.

Al iniciar este trabajo de grado se plantea la importancia de la modularidad en los modelos de procesos de software, tanto por razones de comprensibilidad del modelo, como por la mantenibilidad para posibles futuras modificaciones. En este capítulo se presenta la evaluación del prototipo AVIMO-PS propuesto en este trabajo de grado usando cuatro modelos de procesos de software, dos de procesos industriales y dos de código abierto, en el marco de un estudio de caso.

## 4.1 Metodología

La metodología de este estudio de caso se basó en la guía de Runeson [82] y en las recomendaciones de Yin [83]. En éste estudio de caso se realiza con varias unidades de análisis, que arrojan resultados, los cuales permanecen verdaderos sólo en dichos casos

en específico y que pueden ser replicables en casos similares. No obstante, gracias a este caso de estudio, se logra tener una visión más general, y completa acerca del objeto de estudio, razón por la cual es muy adecuado para nuestro objetivo, ya que permite estudiar los fenómenos contemporáneos en un ambiente real, proveyendo una metodología para observar los atributos y características más significativas e importantes para el estudio. Sin embargo, estos atributos solo se entenderán en su totalidad, una vez se hayan analizado todos los datos simultáneamente, es decir, cuando se estudie el objeto como un todo, considerando más muestras y más mecanismos de evaluación y análisis, como la experimentación por ejemplo.

Para conducir este caso de estudio, se sigue los cinco pasos de la metodología propuesta por Runeson [82]:

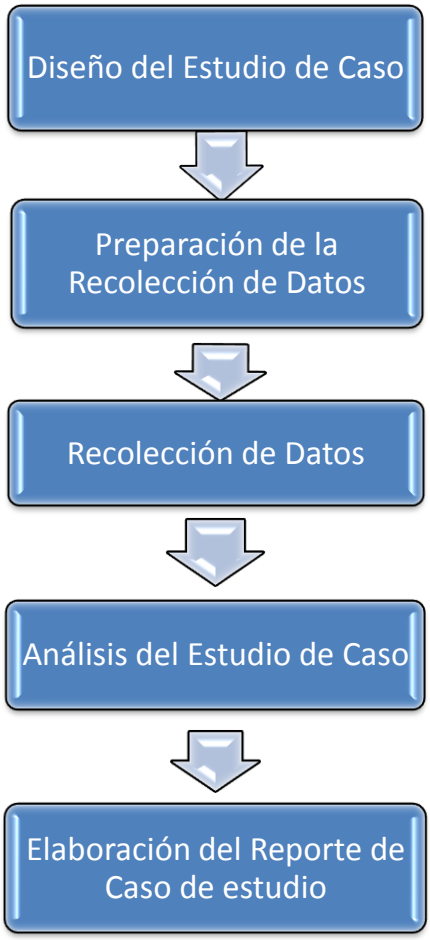



**Figura 25. Metodología del estudio de caso**

- **Diseño del estudio de caso:** en esta parte del estudio de caso se busca ligar, o asociar los datos a recolectar, con las preguntas iniciales del estudio [83]. Este estudio, se diseñó con el fin de saber si AVIMO-PS es una herramienta que permite analizar de manera visual la modularidad de los modelos de procesos de software, de

una manera útil y fácil. Para lograr este objetivo se trabajó con cuatro unidades de análisis, con cuatro modelos de procesos diferentes. Para evaluar la propuesta se plantearon un conjunto de indicadores, con sus respectivas métricas e instrumentos para obtenerlas. Y para la obtención de los datos, se describe los procesos que se utilizan para la recolección de los mismos, así como el reporte de los resultados del caso de estudio.

- **Preparación de la Recolección de Datos:** en el estudio de caso, no solo se especifican los instrumentos, sino cómo se van a utilizar. En este caso, se han diseñado instrumentos para la obtención de los datos, entre ellos tenemos, una capacitación introductoria de la herramienta y conceptos relacionados. Además, se define el protocolo de observación, y los instrumentos para la recolección de los datos, tales como encuestas que son entregadas a los cuatro sujetos de investigación al finalizar la sesión, las planillas de tiempo, y la planilla de errores encontrados, además de una planilla de observaciones, que se llena durante el desarrollo del estudio del caso, para documentar todo los sucesos relevantes o destacados.

- **Recolección de Datos:** La evidencia del estudio de caso, se puede obtener o recolectar mediante documentos, observación participativa, entrevistas, artefactos físicos, y observación directa, como lo menciona Yin [83]. Algunos de las técnicas más utilizadas para la obtención de los datos en las investigaciones cualitativas, y en particular en los estudios de caso, son las entrevista, la observación directa y el análisis de documentos. Todos éstos fueron usados para el desarrollo de éste caso. La obtención de los datos se realizó principalmente a través de la observación directa realizada a los sujetos de la investigación, los resultados de la actividad y de las encuestas entregadas a los mismos.

- **Desarrollo del caso:** el estudio de caso se llevó a cabo en dos sesiones de 5 horas aproximadamente, con dos sujetos de investigación en cada una. Primero se realizó una presentación con los conceptos relacionados a la cohesión, acoplamiento e inestabilidad así como también el manejo de la herramienta AVIMO-PS, para la importación de los modelos de procesos de software, y la generación de las visualizaciones. Después de ello se asignó un modelo de proceso a cada uno de los participantes, y se hace entrega de una plantilla para el reporte de errores, en dicha plantilla se pueden hacer anotaciones de observaciones de usabilidad, errores en la aplicación o posibles errores o anomalías percibidas en el modelo de proceso. Una vez se termina con la introducción y asignación del modelo de proceso, cada ingeniero de proceso, procede a iniciar con el análisis de los modelos de proceso utilizando la herramienta AVIMO-PS, paralelo a ello, se lleva a cabo el protocolo de observación por parte del equipo de investigación. Algunas de las evidencias fotográficas obtenidas durante la ejecución del estudio de caso, se muestran en esta sección, con previa autorización de los sujetos de investigación, sobre el uso de derechos de imagen sobre fotografía.

- **Análisis del Estudio de Caso:** en esta parte de análisis de los datos, se busca estudiar, agrupar, tabular, o recombinar la información, con el fin de lograr alcanzar el propósito del estudio de caso. Dicho análisis se realizó a nivel cualitativo y cuantitativo, partiendo de los datos que se obtuvieron de las encuestas, plantillas, y de la observación realizada hacia los sujetos de investigación. A partir de las encuestas y plantillas, se logró obtener información sobre sugerencias de mejora de rendimiento y usabilidad de la herramienta.

- **Elaboración del Reporte de Caso de estudio:** el reporte de los resultados de este estudio de caso, se realizó de manera escrita y analítica, y se presenta en este documento. Siendo de interés para la audiencia en general del entorno de ingeniería de proceso de software.

### 4.1.1 Pregunta de Investigación

El prototipo AVIMO-PS se ha implementado pensando en la tarea de evaluación de los modelos de proceso de software, y la identificación visual de posibles anomalías sobre los modelos de proceso; además conocer si ha sido definido adecuadamente, para facilitar evaluar la modularidad de los procesos, de tal manera, que facilite la modificabilidad durante su evolución. Con esto se busca facilitar a los ingenieros de procesos detectar problemas y analizar datos en modelos definidos de manera temprana, utilizando la capacidad humana para visualizar e interpretar las vistas, logrando evaluar la coherencia y confiabilidad de un proceso de software. Esto conlleva a formular la siguiente pregunta para el estudio de caso ¿Resulta AVIMO-PS una solución útil y fácil para evaluar la modularidad de los modelos de proceso de software especificados en EPFC? La respuesta de esta pregunta ayuda a complementar la pregunta de investigación del presente trabajo de grado ¿Cómo identificar visualmente si un modelo de proceso de software ha sido modularmente bien para facilitar su cambio y evolución?

## 4.2 Objetivo del Estudio de Caso

El objetivo de este estudio de caso es someter a evaluación cuatro modelos de proceso, dos de tipo empresarial y dos de tipo comunidad (código abierto), estudiando los resultados con cuatro ingenieros de sistemas que tienen experiencia en formulación de procesos con SPEM2.0 en EPFC, y con ello identificar al menos un patrón problemático asociado a la modularidad.

## 4.3 Selección del Estudio de Caso

Para la selección de los estudios de caso, se siguieron varios criterios, dentro de ellos se tienen:

- **Disponibilidad de los procesos en EPFC:** no todos los modelos procesos se encuentra especificados formalmente, tampoco todos los especificados formalmente lo han hecho en SPEM2.0 y EPFC y son mucho menos los que las empresas facilitan debido a la confidencialidad que se requiere. Por ello, este trabajo de grado se seleccionó los modelos de proceso, de los pocos disponibles de la comunidad y de la industria.
- **Disponibilidad de los ingenieros de procesos:** La tarea de definir y evaluar los modelos de procesos de software es desarrollada por un ingeniero de procesos, siendo ellos los indicados para evaluar la complejidad y practicidad de nuestra propuesta.
- **Revelatorio[17]:** de los pocos modelos de proceso disponibles, no todos son aptos para evaluar la capacidad de AVIMO-PS, particularmente se buscan procesos que estén estructurados en varios paquetes de contenido de método, para lograr evaluar empíricamente las capacidades de AVIMO-PS.
- **Típico[18] (casos reales en la industria):** se han tenido en cuenta modelos de procesos en dos ámbitos bien diferenciados: modelos de proceso open source y modelos de proceso propietarios. También se han tomado modelos de proceso correspondientes a modelos evaluados con un estándar de calidad y modelos de proceso definidos de forma independiente a cualquier estándar. Así se cubren diferentes tamaños, propósitos, complejidades y contextos.

## 4.4 Descripción del Estudio de Caso

El estudio de caso es de tipo embebido [83], con cuatro unidades de análisis (cuatro actividades de evaluación por pares), correspondientes al análisis de cuatro modelos de proceso distintos con cuatro ingenieros de procesos diferentes: dos modelos de proceso open source y dos modelos de proceso industriales. Y de los mismos cuatro procesos, dos evaluados mediante estándares (CMMI, ISO 9000) y dos independientes de una implementación de un estándar.

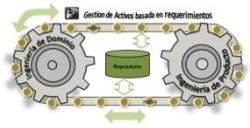
Con el fin de realizar una evaluación empírica del prototipo AVIMO-PS, se seleccionaron como sujetos de evaluación a cuatro ingenieros de procesos con conocimientos en la especificación SPEM2.0 y evaluación de procesos. Con respecto a la formación de los sujetos de investigación, los cuatro tienen el conocimiento necesario en SPEM2.0 para interpretar y diseñar modelos de proceso, así como también poseen experiencia en adaptación y mejora de los procesos. Dos de los sujetos de investigación tienen un conocimiento básico en el manejo de la herramienta AVISPA, lo cual ayuda a tener una

[17] Un caso de estudio revelatorio existe cuando un investigador tiene la oportunidad de observar y analizar un fenómeno previamente inaccesible a la investigación científica.
[18] Un caso de estudio típico tiene como objetivo capturar las circunstancias y condiciones de situaciones comunes.

mayor fluidez al utilizar AVIMO-PS. Los otros dos ingenieros de proceso carecen de éstos conocimientos previos.

### 4.4.1 Los modelos de procesos de desarrollo del caso

Para desarrollar este estudio de caso, se utilizan 4 modelos de proceso de software, los cuales se encuentran especificados formalmente en SPEM2.0 Y EPFC. Dos de estos modelos, SmallSPL, Tutelkan son de fuente abierta, y los modelos Rhiscom y Amisoft son procesos industriales que no se encuentran disponibles para la comunidad en general.

| | Fuente abierta | Fuente cerrada |
|---|---|---|
| **Sin implementar algún estándar** |  **SmallSPL** | rhiscom® |
| **Basado en CMMI o ISO 9001** | Tutelkan® El Software de la Región del Maule www.tutelkan.com | AmisOft Soluciones y Servicios TI |

Small SPL, es un proceso de desarrollo que pretende facilitar la adopción del paradigma de las Líneas de Productos Software - (Software Product Lines - SPL) como una estrategia de producción en las Mi pymes desarrolladoras de software de la región, esta estrategia busca reducir costos, a través de la reutilización planificada y sistemática de los productos software.
Rhiscom[19] ha proveído de satisfactorios resultados en distintas áreas, integrando el Front-end y Back-office apoyado de soluciones propias a la medida, a grandes cadenas en Chile y fuera de sus fronteras. Además ha institucionalizado en cada uno de sus procesos el uso de la metodología RHUP (Rhiscom Unified Process). Esta metodología se centra en las necesidades de sus clientes, las buenas prácticas de la industria del software y estándares definidos con soporte de tecnologías modernas.

Tutelkan[20], es una empresa desarrolladora de software de la Santiago de Chile, que ha definido un proceso de software en particular, cuyas partes pueden ser reutilizadas y modificadas para crear nuevos procesos de software. Este proceso contiene prácticas probadas por PyMEs de software chilenas, y está alineado los modelos CMMI ML2 y ML3, e ISO 9001:2000.

El proceso organizacional de Amisoft[21] tiene en cuenta, en principio dos tipos de proyectos: mantenimiento y desarrollo. Amisoft es una empresa de servicios de

[19] http://www.rhiscom.com/
[20] http://www.tutelkan.info/
[21] http://www.amisoft.cl/

tecnología de información localizada en Santiago de Chile, la cual diseñó e implementó su propio proceso para el desarrollo y mantención de software llamado APF (Amisoft Process Framework) el cual implementa las prácticas de CMMI e ISO 9001 y se basa en la gestión de proyectos e ingeniería de software reconocidas por la industria de software mundial.

## 4.5 Indicadores y Métricas

Para obtener una evaluación de éste estudio de caso, es necesario establecer métricas e indicadores, con el fin de establecer un correspondencia entre un dominio empírico (mundo real) con un mundo formal, matemático.

La propuesta de este trabajo de grado busca identificar errores asociados a la modularidad de modelos de procesos de software, utilizando la capacidad visual de los ingenieros de proceso, por lo tanto, la herramienta AVIMO-PS debe facilitar la identificación de errores reales, y en la medida de lo posible disminuir los falsos positivos, contribuyendo en su efectividad, siendo la efectividad la capacidad de la herramienta de facilitar al usuario alcanzar objetivos con precisión y completitud. Para ello debe generar visualizaciones intuitivas, poco complejas y fáciles de comprender.

Es por ello que se han definido algunos indicadores y métricas, las cuales ayudan a ponderar de una forma objetiva las cualidades que consideramos más importantes en nuestra propuesta, como la efectividad, complejidad, comprensión y usabilidad, las cuales se muestran en la Tabla 1.

| Objetivo | Indicadores | Métricas | Instrumentos |
|---|---|---|---|
| ¿Es AVIMO-PS una solución útil y fácil para evaluar la modularidad de los modelos de proceso de software | Efectividad | Porcentaje de errores reales respecto los errores detectados, calculado con la Fórmula 14. | AVIMO-PS, EPFC Reporte de Errores |
| | Complejidad | Complejidad percibida por los Ingenieros de Procesos con respecto a AVIMO-PS. | Encuesta Protocolo de Observación |

| | | | |
|---|---|---|---|
| especificados en EPFC? | Comprensión | Nivel de comprensión obtenido por los ingenieros de procesos con respecto a AVIMO-PS. | Encuesta Protocolo de Observación |
| | Usabilidad | Usabilidad de la herramienta (Blueprints) de acuerdo la experiencia de los Ingenieros de Procesos con otras tecnologías de modelado. | Test de Usabilidad SUMI[22] |

**Tabla 1. Métricas e Indicadores**

A continuación se expone de forma detallada las métricas e indicadores del estudio de caso.

- **Efectividad:** En el momento en el que los participantes evalúen los modelos de procesos con AVIMO-PS, van reportando los errores encontrados. Esta métrica mide el porcentaje de errores reales, respecto a los reales potenciales encontrados por AVIMO-PS.

$$\boldsymbol{Efectividad = \frac{ER * 100}{EE}}$$

**Fórmula 15. Indicador de Efectividad**

Donde:
**EE** son los errores encontrados por los evaluadores
**ER** son los errores reales encontrados, después de ser verificados por los responsables de los modelos.

- **Complejidad:** Cuando los usuarios (sujetos de investigación) interactúen con el prototipo AVIMO-PS, mientras evalúan la modularidad de los modelos de procesos de software, pueden suministrar información a través de encuestas, o protocolos de observación, que será recopilada con el fin de medir la complejidad del prototipo. Para ello se ha definido la Fórmula 16, para calcular la complejidad percibida:

$$\boldsymbol{Cplj = \frac{\sum_{i=1}^{n} RS_i}{n}}$$

**Fórmula 16. Indicador de Complejidad**

Donde:
***Cplj*** Es el nivel de complejidad de uso del prototipo AVIMO-PS
$\mathbf{RS_i}$ Es la respuesta seleccionada por el encuestado a la pregunta *i.* Cuyo valor está entre uno y cinco
$\mathbf{n}$ Es el número de preguntas encuestadas asociadas a la complejidad.

- **Comprensión:** Cuando los usuarios (sujetos de investigación) analicen la información propia de los Blueprints de los modelos de proceso de software

[22] http://www.ucc.ie/hfrg/questionnaires/sumi/index.html

generados con el prototipo AVIMO-PS, de tal forma que permitan evaluar la modularidad de los modelos de procesos, para ello se recopila la información a través la encuesta. Para ello se ha definido la Fórmula 17, para calcular la facilidad percibida:

$$Cpsn = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{RS_i}}{\mathrm{n}}$$

**Fórmula 17. Indicador de Comprensión**

Donde:

***Cpns:*** Es el nivel de comprensión de los Blueprints generados con AVIMO-PS

$\mathbf{RS_i}$ Es la respuesta seleccionada por el encuestado a la pregunta *i.* Cuyo valor está entre uno y cinco

**n** Es el número de preguntas de la encuesta asociadas a la comprensión.

- **Usabilidad[23]:** Podemos medir la usabilidad por parte de los usuarios (sujetos de investigación) al momento de interactuar con el prototipo AVIMO-PS, con el fin de obtener una retroalimentación acerca de esfuerzo realizado para utilizar el prototipo, asi mismo,se puede conseguir información de objetivos específicos con efectividad, eficiencia y satisfacción en el contexto definido**.** Por lo anterior se ha elaborado Fórmula 18, , para calcular la usabilidad percibida:

$$\boldsymbol{Usbl} = \frac{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{RS_1}}{\mathbf{n}}$$

**Fórmula 18. Indicador de Usabilidad**

Donde:

***Usbl*** es el esfuerzo requerido por los usuarios para utilizar el prototipo denominado como.

$\mathbf{RS_i}$ Es la respuesta seleccionada por el encuestado a la pregunta *i.* Cuyo valor está entre uno y cinco

**n** Es el número de preguntas de la encuesta.

Una de las formas más efectivas de medir la usabilidad es utilizando cuestionarios "tipo test" diseñados para tal propósito. En dichos test se deben contestar una serie de preguntas, para las cuales existe un determinado rango de respuestas. El principal motivo para la realización de estos cuestionarios, está en que permiten recolectar respuestas concretas, proporcionando datos comprobables mediante, por ejemplo, estudios estadísticos. Para la realización de estas pruebas se utilizó el esquema planteado por SUMI (The Software Usability Measurement Inventory) [84] que se encuentra entre uno de los más relevantes [85].

## 4.6 Ejecución del Estudio de Caso

[23] http://iso25000.com/index.php/iso-iec-9126.html

Para desarrollar el estudio de caso, se trabajó con cuatro ingenieros de procesos, responsables cada uno de un modelo de proceso, utilizando como método de trabajo la revisión por pares [86]. Para ello se siguieron las siguientes tareas:

1. **Entrenamiento:** los participantes fueron entrenados en los conceptos relacionados con la modularidad de los modelo de proceso, el manejo de la herramienta AVIMO-PS, y de la metodología de evaluación seguir. En la Figura 26, se muestra el momento en que se realiza la capacitación introductoria que abarca los conceptos necesarios para llevar a cabo el análisis de los modelos de procesos.

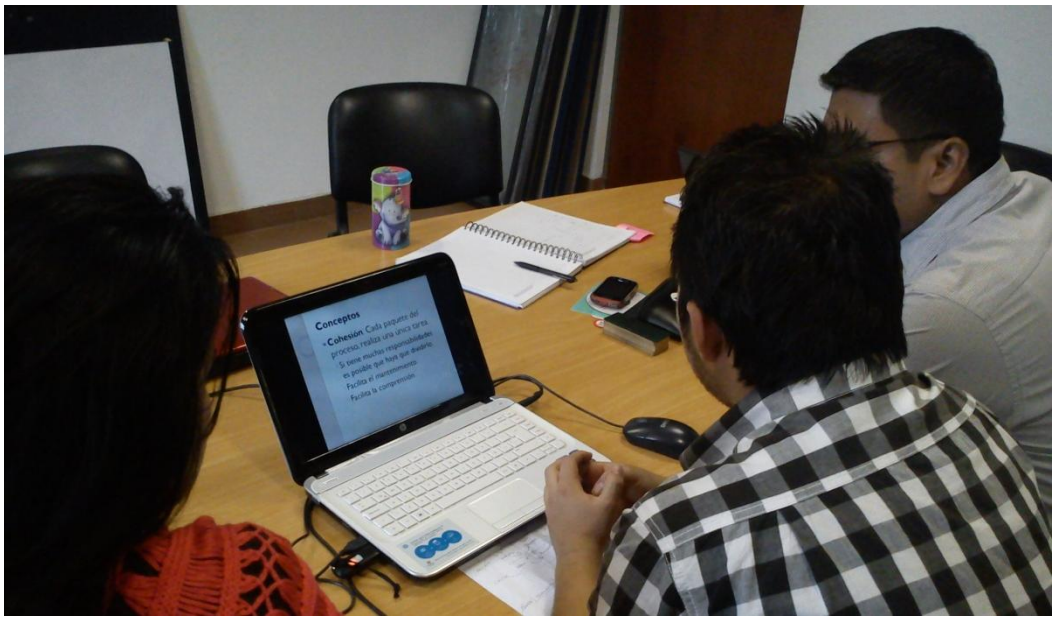



**Figura 26. Capacitación introductoria**

2. **Preparación de la revisión:** se adecuó la aplicación y el espacio de trabajo, se asignaron evaluadores a cada modelo de proceso, posteriormente se les entregó la planilla de reporte de errores y el proceso a evaluar en formato XML.

3. **Desarrollo de la evaluación:** los ingenieros de proceso evaluadores utilizando AVIMO-PS evaluaron el modelo de proceso asignado, siguiendo un enfoque de inspección, en donde fueron reportando los errores en la planilla de reporte de errores. Durante dicho proceso de evaluación los investigadores siguieron un protocolo de observación.

**Figura 27. Sujetos de investigación en la primera sesión**

**Figura 28. Sujetos de investigación en la segunda sesión**

4. **Revisión de la evaluación:** los ingenieros de proceso responsables de su respectivo proceso, contrastan con el evaluador del proceso los errores reportados para determinar si efectivamente son problemas reales del modelo de proceso. Durante esta tarea se actualiza la planilla de reporte de errores.
5. **Entrega de resultados:** los ingenieros de proceso entregan los reportes de errores resultantes a los investigadores y se socializa el ejercicio.
6. **Aplicación de encuestas:** terminado el ejercicio los investigadores aplicaron las encuestas a los ingenieros de proceso en su rol de evaluadores, con el fin de obtener información sobre el enfoque de evaluación, complejidad y facilidad de uso de la herramienta AVIMO-PS.

## 4.7 Resultados

El análisis se centra en tres posibles errores: paquetes altamente acoplados, paquetes con muy baja cohesión y paquetes altamente inestables. A continuación se presentan los principales resultados cuantitativos y cualitativos.

### 4.7.1 Resultados Cuantitativos

**Mediciones directas:**

A continuación se muestra los datos que se obtuvieron en el desarrollo del estudio de caso, en donde los sujetos investigados recibieron la capacitación, y realizaron el análisis visual de un modelo de proceso de software. La medida de tiempo ha sido tomada en horas. En la Tabla 2 se expone la asignación de los modelos de proceso a los sujetos de investigación, además la Tabla 3 muestra la información del tiempo que cada unidad utilizó para analizar los modelos de proceso de software.

| Sujeto | Modelo de proceso de software | Cantidad de Paquetes |
|---|---|---|
| **Sujeto 1** | Tutelkan | 26 |
| **Sujeto 2** | Rhiscom | 7 |
| **Sujeto 3** | Small SPL | 3 |
| **Sujeto 4** | Amisoft | 20 |

**Tabla 2. Asignación de modelos de procesos de software a los sujetos de investigación**

| Actividad | Unidad de Análisis (Tiempo en horas) | | | | |
|---|---|---|---|---|---|
| | **Sujeto 1** | **Sujeto 2** | **Sujeto 3** | **Sujeto 4** | **Promedio** |
| **Entrenamiento** | 0.3 | 0.4 | 0.5 | 0.5 | 0.425 |
| **Entendimiento del Modelo de proceso** | 0.15 | 0.25 | 0.1 | 0.5 | 0.25 |
| **Análisis del modelo de proceso** | 2.1 | 2.0 | 1.0 | 1.8 | 1.75 |
| **Totales** | **2.55** | **2.65** | **1.6** | **2.8** | **2.4** |

**Tabla 3. Registro del tiempo empleado por los ingenieros de procesos durante la sesion**

La cantidad de posibles errores encontrados, gracias al análisis visual con AVIMO-PS, se relacionan en la Tabla 4. Los cuales se detallan en la Tabla 7.

| | Efectividad de AVIMO-PS | | | | |
|---|---|---|---|---|---|
| | **Sujeto 1 (Tutelkan)** | **Sujeto 2 (Rhiscom)** | **Sujeto 3 (Small SPL)** | **Sujeto 4 (Amisoft)** | **Total** |
| **Número de posibles anomalías encontradas** | 13 | 5 | 1 | 10 | 29 |
| **Numero Anomalías efectivas encontradas** | 11 | 4 | 1 | 9 | 25 |
| **Efectividad de la herramienta** | 84,6% | 80% | 100% | 90% | 88,65% |

**Tabla 4. Número de posibles anomalías encontradas**

De acuerdo a las encuestas que se entregaron al finalizar la actividad de análisis de los modelos de proceso por parte de los ingenieros de procesos, se ha logrado analizar los datos obtenidos y con ello lograr calcular un valor cuantitativo de la complejidad, comprensión, y usabilidad de la herramienta AVIMO-PS, de una manera más objetiva. Las preguntas realizadas en la encuesta, asociadas a la complejidad y comprensión se encuentran en el Anexo B: ENCUESTA PARA MEDIR LA COMPLEJIDAD Y COMPRENSIÓN; Y las preguntas relacionadas con la usabilidad se encuentran en el Anexo C: TEST DE USABILIDAD.

| | Indicadores de Complejidad, Comprensión y Usabilidad de AVIMO-PS | | | | | |
|---|---|---|---|---|---|---|
| | **Sujeto 1** | **Sujeto 2** | **Sujeto 3** | **Sujeto 4** | **Promedio** | **Promedio %** |
| **Complejidad** | 1,6 | 1,8 | 1,4 | 1,6 | 1,6 | 15,00% |
| **Comprensión** | 4,4 | 4,6 | 4,4 | 4,6 | 4,5 | 87,50% |
| **Usabilidad** | 4,6 | 4,5 | 3,9 | 2,9 | 4,0 | 75,00% |

**Tabla 5. Indicadores de Complejidad, Comprensión y Usabilidad de AVIMO-PS**

Como se observa en la Tabla 5, y confrontando con la Fórmula 18. Indicador de Usabilidad, en el cual el valor entre más cercano sea a 1, el grado de usabilidad es del 0%; También se puede observar que el nivel de complejidad para los sujetos investigados resulta muy **bajo** y que el nivel de satisfacción con respecto a la comprensión y facilidad de uso reflejado por ellos es **alto**. Esto infiere que la complejidad de las vistas junto con sus representaciones gráficas de las métricas, no afecta sobremanera la complejidad percibida por los ingenieros de proceso. Aunque la complejidad de los Blueprints y la carga visual aumenta un poco, ésta no es significativamente percibida por los ingenieros de procesos, además permite realizar dicho análisis de una manera más intuitiva para el usuario. Por otro lado, el beneficio resulta visible en comparación con el análisis de la modularidad de los procesos de software especificados con SPEM2.0 sin AVIMO-PS.
Cabe resaltar que al inicio de la sesión, la usabilidad no fue igualmente percibida por los distintos sujetos, los sujetos expertos en AVISPA, quienes ya están acostumbrados a su metáfora de interface de usuario de Moose, tenían una mayor agilidad, para navegar en la herramienta, sin embargo los ingenieros sin experiencia, a pesar que al inicio encontraron

más dificultades en usar la herramienta, al transcurrir la sesión, fueron superando los obstáculos. En el resultado de la encuesta de usabilidad de la herramienta, indica que a pesar de que hay aspectos considerables a ser mejorados en AVIMO-PS y en general en AVISPA, la herramienta cuenta con una interfaz diciente y que permite un aprendizaje relativamente rápido.

### 4.7.2 Resultados cualitativos

#### 4.7.2.1 Apreciaciones de los ingenieros de procesos

En el desarrollo del estudio del caso, uno de los sujetos de investigación hizo una apreciación muy valiosa sobre los Blueprints: *"el tener todos los paquetes de contenido de método de manera gráfica y simplificado en el mismo espacio de trabajo, le facilita la comprensión del proceso de software y le permite encontrar falencias de manera rápida"*. Otro de los sujetos señaló que, "*si se carga un proceso más complejo, es decir, con mayor numero de paquetes de contenido de método, número de tareas, productos de trabajo, mayor números de roles e interacciones (si el proceso es mucho más grande), el Blueprint generado sería más pesado visualmente, dificultando el análisis visual"*, sin embargo los procesos utilizados en éste estudio no fueron pequeños. Uno de los sujeto de investigación después de realizar el análisis del modelo asignado, siguiere que se utilice un sistema de colores del semáforo para destacar los errores. A pesar de que los sujetos investigados tienen un conocimiento heterogéneo, los cuatro coincidieron en que representar el acoplamiento, cohesión e inestabilidad de una manera gráfica es más fácil de interpretar. De todas formas el usuario debe recibir una preparación de los evaluadores en conceptos como acoplamiento, cohesión e inestabilidad, y por otro lado, se debe incorporar algunos elementos de lectura e interpretación en las vistas de AVIMO-PS.

Para ver los resultados de la encuesta a los ingenieros de proceso diríjase al Anexo C.

#### 4.7.2.2 Apreciaciones de los Investigadores durante el desarrollo del caso

Durante el desarrollo de este estudio de caso, observamos que los sujetos de investigación tienen muy pocos problemas al utilizar la herramienta AVIMO-PS, ya que en la capacitación se enseña cómo cargar los procesos y cómo generar los diferentes Blueprints. Además algunos de ellos poseen experiencia en el manejo de la herramienta AVISPA, así como también, conocimientos en modelamiento de procesos de software. Sin embargo se observa que hay cierta dificultad en entender los modelos de proceso que se analizan por primera vez. Pero esta barrera la superan, gracias a la experiencia que han tenido con otros procesos similares, aunque uno de los sujetos de investigación tuvo la necesidad de ver el modelo de proceso en la herramienta EPFC, con el fin de poder entender un poco más a fondo la estructura del modelo. También se observa que cuando el sujeto de investigación comienza el análisis de un proceso, el avance es un poco lento en los primeros minutos, esto es porque no conocen a fondo el proceso, pero a medida que avanza la sesión, la detección de posibles errores y las sugerencias de mejora van apareciendo.

En ocasiones, un evaluador del modelo, al detectar una anomalía lo socializa con los demás para obtener observaciones o complementar la información sobre dicho problema. Por ejemplo al generar los Blueprints enfocados en los roles o artefactos, estos sólo se componían de un paquete de contenido de método, en donde agrupan todos los elementos, a primera vista esto podría confundir o indicar que es un problema, pero al socializar la información se concluye que muchos modelos de procesos agrupan todos los roles, o artefactos en un solo paquete de contenido de método. Este caso fue recurrente en todos los modelos de proceso estudiados. En particular, a uno de los sujetos de investigación se le asignó el modelo de proceso APF, siendo este un proceso conocido para el evaluador, lo que le ayudó a realizar el análisis de una manera más rápida. Por otro lado el modelo Rishcom, fue asignado a otro evaluador, el cual era desconocido para él, pero esto no fue un impedimento para llevar a cabo la evaluación.

Cuando los evaluadores iniciaron el análisis de los Blueprint de Inestabilidad, fue necesario reforzar los conceptos asociados a este Blueprint, tales como el acoplamiento y acoplamiento aferente, conceptos que dan lugar a la inestabilidad. Aún así, teniendo en cuenta la corta capacitación sobre los conceptos de acoplamiento, cohesión e inestabilidad, y sobre el funcionamiento del prototipo AVIMO-PS, la percepción de los ingenieros de proceso en el momento de realizar el análisis de los modelos fue muy buena, ya que pudimos apreciar que no se percibió frustración al utilizar la herramienta para el análisis, por ello, concluimos que el prototipo presenta una usabilidad aceptable, sin embargo como lo expresó uno de los ingenieros, es posible que para modelos más grandes el tema de la navegabilidad dentro del espacio de trabajo pueda complicarse.

## 4.8 Análisis de Resultados

### 4.8.1 Análisis de Resultados Cuantitativos

En la Tabla 3Tabla 3. Registro del tiempo empleado por los ingenieros de procesos durante la sesion , se muestran los tiempos en horas necesarias para ejecutar las actividades, en promedio el tiempo de entrenamiento para utilizar la herramienta y la explicación de los conceptos necesarios para llevar a cabo el estudio de caso, tardó 0.425 personas-hora, y el entendimiento del modelo de proceso en promedio tiene un esfuerzo de 0.25 personas-hora. Estos tiempos, son tiempos razonables que una organización con foco en sus procesos, perfectamente puede patrocinar para la evaluación de procesos de software para el continuo mejoramiento, y por tanto requieren evolucionar, conocer el valor de los procesos como determinantes de la calidad de los productos y la productividad de los proyectos.

Una medida que permite visualizar, el grado de correctitud en que los sujetos investigados realizaron el proyecto de análisis visual de los modelos de proceso de software, es la

efectividad, permitiendo estudiar el grado en que se cumple el objetivo de la herramienta AVIMO-PS, que es lograr un análisis visual de la modularidad de los modelos de procesos de software, y poder detectar de manera temprana problemas en el diseño del modelo, y con ello responder a la pregunta de investigación de este trabajo de grado, que es, ¿Cómo identificar visualmente si un modelo de proceso de software ha sido modularmente bien definido para facilitar su cambio y evolución?
Como muestra la Tabla 4, en promedio, la efectividad de AVIMO-PS para detectar problemas fue del 88,65%, con lo cual, podemos decir que es bastante alta, respecto a los reportes de AVISPA y respecto a su utilidad. Si bien se esconden falsos positivos, el conjunto de problemas reduce sustancialmente el conjunto de elementos a analizar, lo que es una ganancia para el verificador de procesos. Sin embargo los posibles problemas detectados, deben de someterse a un riguroso análisis, para poder determinar si se trata de un falso positivo o si realmente corresponde a una anomalía en el proceso, para ello cada posible problema es puesto en contexto y estudiado por los ingenieros de proceso encargados.

Según los resultamos expuestos en la Tabla 5, en donde el rango de los valores está entre 1 y 5, podemos concluir que en promedio, las unidades de análisis evidenciaron tener un 87,5% de comprensión de los Blueprints generados por AVIMO-PS, consideramos que es un valor alto, ya que el análisis visual de modelos de procesos de software es una técnica nueva dentro del diseño de procesos software. Por otro lado, el resultado del nivel de complejidad percibido por los sujetos de investigación, es de 15%, en conclusión, según nuestras convenciones, se llega a que los sujetos están *"Totalmente en desacuerdo",* con que la complejidad del sistema es alta.

El grado de facilidad y entendimiento de AVIMO-PS, se obtuvo mediante la medida de la usabilidad, que para calcularla utilizamos los resultados arrojados por el test de usabilidad respondido por los sujetos investigados, que demuestran que en promedio el 87,5% de las respuestas, concluyen estar "Totalmente de Acuerdo" con que la herramienta tiene adecuada usabilidad. Esto es muy significativo para nosotros, ya que todos los sujetos de investigación utilizaban AVIMO-PS por primera vez, y para la mitad fue su primera experiencia analizando modelos de proceso de software de manera gráfica. En la encuesta realizada de usabilidad, no incluimos todos los atributos, conceptos asociados al proceso de usabilidad. Sólo tenemos en cuenta los aspectos más importantes para este estudio de caso, como lo son, la utilidad de tener todas las representaciones gráficas asociadas a las métricas, la navegación en el Blueprint, además de poder generar rápidamente las visualizaciones de los modelos y la facilidad de aprendizaje percibida por los sujetos de investigación utilizados en el estudio de caso.

### 4.8.2 Análisis de Resultados e Identificación del Patrón de Error

Durante este estudio de caso AVIMO-PS fue utilizado con el fin de encontrar problemas u oportunidades de mejora en cuatro modelos de proceso de software, apoyándose en la

hipótesis de que una herramienta visual ayuda al ingeniero de procesos, para analizar de una manera efectiva los modelos de proceso de software.
AVIMO-PS se creó de manera incremental y se realizaron las pruebas inicialmente con un test que representaba un modelo de proceso de software muy pequeño, pero para poder evaluar el enfoque de la herramienta, fue necesario modelos de procesos reales. Por esta razón se seleccionaron los modelos de proceso especificados formalmente en EPCF. La experiencia que se obtuvo al validar las métricas propuestas en este trabajo de grado aplicadas estos cuatro modelos de proceso, arrojaron un listado de posibles problemas en los modelos, los cuales fueron analizados en detalle para poder identificar si correspondían a oportunidades de mejora reales para los procesos o se trataban de falsos positivos. Luego de analizar todos los resultados estos fueron discutidos con cada uno de los ingenieros de proceso.

Para el análisis de cada uno de los modelos de proceso de software, cada sujeto de investigación siguió los siguientes pasos:

**Importación*:*** el sujeto de investigación tiene la posibilidad de importar el modelo a Eclipse Process Framework Project (EPF), con el fin de realizar una revisión rápida y poder ver algunas descripciones. Después de ello importa el modelo de proceso en AVIMOP-PS, una vez cargado el modelo, la herramienta está lista para proceder a generar los Blueprints.

**Visualización***:* el sujeto de investigación realizaba un análisis visual exhaustivo a cada Blueprint, con el fin de encontrar anomalías, cada vez que se detectaba una de ellas, se registraba en una planilla de errores, el nombre del paquete, el tipo de problema (cohesión, acoplamiento o inestabilidad) y una descripción del posible problema, esto con el fin de realizar una futura inspección y poder determinar si se trata de un error real o un falso positivo.

**Figura 29. Análisis de los Blueprint durante la primera sesión**

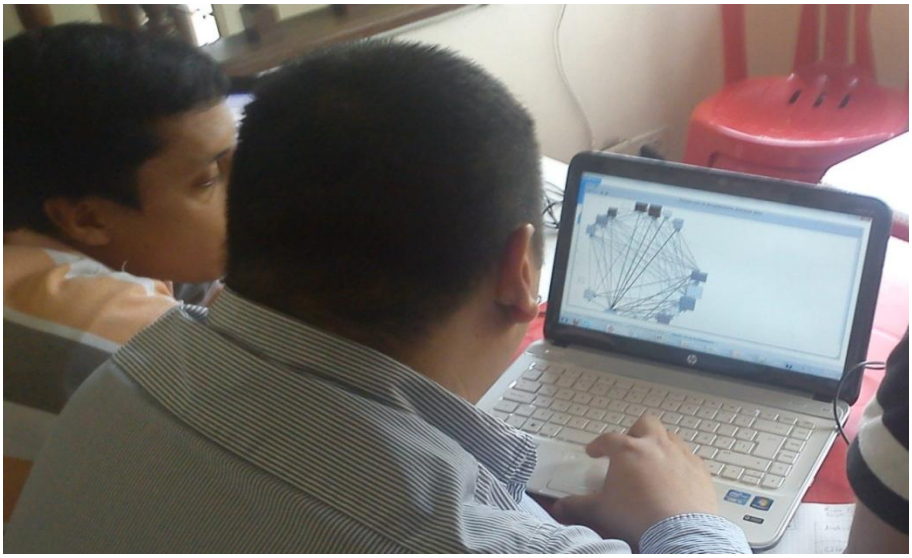


**Figura 30. Análisis de los Blueprint durante la segunda sesión**

En la Tabla 6 se exponen el número de problemas encontrados en los cuatro modelos de procesos durante la inspección de los Blueprint.

| Blueprint | Problema Identificado | Cantidad de veces encontrado |
|---|---|---|
| **Acoplamiento y Cohesión** | Paquetes con baja cohesión, paquetes con alto acoplamiento, paquetes con elementos sin relaciones internas. | 16 |
| **Inestabilidad** | Paquetes Inestables, elementos con alto acoplamiento hacia otros paquetes. | 9 |
| **Paquetes Aislados** | Paquetes aislados, sin ningún tipo de interacción con los demás paquetes del modelo de proceso. | 4 |

Tabla 6. Problemas encontrados en los modelos de proceso

**Análisis de los problemas:** Una vez que el ingeniero de proceso detecta un problema potencial en el modelo, lo sometía a un análisis para poder determinar si se trataba de un error como tal, si era el caso, lo socializaba con el otro ingeniero de proceso, que también participaba en la sesión pero con otro modelo de proceso. Una vez realizada la discusión, se confirma, descarta o se apunta el problema en la planilla de errores. En la Figura 31, se muestra, los sujetos de investigación socializando algunas dudas que tenían sobre los modelo, con el fin de aclarar si se trataba de un problema en el modelo de proceso o era un error en la interpretación de los Blueprint.

**Figura 31. Socialización al detectar una posible anomalía**

**Recolección de datos:** la recolección de los datos, se realizó en dos sesiones con una duración aproximada de 5 horas, y dos ingenieros de proceso en cada una. Durante cada sesión se recolectaban datos mediante la observación directa, así como también los potenciales errores encontrados en los modelos y las observaciones realizadas por parte de los sujetos de investigación. La Figura 32, muestra cómo se realizaba en algunas ocasiones la observación directa, para analizar cómo era la experiencia de usuario con la herramienta, o cómo realizaban el estudio de los Blueprints.

**Figura 32. Observación directa, hacia los sujetos de investigación**

**Análisis de los datos:** los datos que se obtuvieron después de las sesiones de análisis de los modelos de proceso realizados por los sujetos de investigación, se estudian en esta sección y se exponen los resultados más significantes.

Los modelos de proceso Tutelkan, Rhiscom, Small SPL, Amisoft, tienen 26, 7, 3 y 20 paquetes de elementos de contenido de método respectivamente, los cuales después de ser analizados, se encontraron algunas posibles anomalías o posibilidades de mejora, en donde cada uno fue sometido a un riguroso análisis, y contrastado entre el evaluador y el dueño del proceso, para determinar si efectivamente se trata de un problema real en el modelo de proceso. El resultado de este análisis se resume a continuación.

| Nombre del proceso | Problema identificado | Paquete | Estado |
|---|---|---|---|
| **Tutelkan** | Baja cohesión | Medición y Análisis | Comprobado |
| **Tutelkan** | Baja cohesión | Planificación del Proyecto | Falso positivo |
| **Tutelkan** | Baja cohesión | Pruebas | Comprobado |
| **Tutelkan** | Baja cohesión | Gestión de Riesgos | Comprobado |
| **Tutelkan** | Baja cohesión | Desarrollo de Requerimientos | Comprobado |
| **Tutelkan** | Baja cohesión | Análisis y Diseño | Comprobado |
| **Tutelkan** | Alto acoplamiento | Definición de procesos organizacionales | Comprobado |
| **Tutelkan** | Paquete aislado | Mejoramiento de procesos organizacionales | Comprobado |
| **Tutelkan** | Paquete aislado | Evaluación formal de decisiones | Comprobado |
| **Tutelkan** | Inestabilidad | Definición de procesos organizacionales | Comprobado |
| **Tutelkan** | Inestabilidad | Desarrollo de requerimientos | Falso Positivo |
| **Tutelkan** | Inestabilidad | Gestión de riesgos | Comprobado |
| **Tutelkan** | Inestabilidad | Administración integrada del proyecto | Comprobado |
| **Rhiscom** | Alto acoplamiento | Comercial | Falso Positivo |
| **Rhiscom** | Baja cohesión | Comercial | Comprobado |
| **Rhiscom** | Baja cohesión | Implementación | Comprobado |
| **Rhiscom** | Inestabilidad | Requisitos | Comprobado |
| **Rhiscom** | Inestabilidad | Comercial | Comprobado |
| **Amisoft** | Alto acoplamiento | Procesos Amisoft | Falso Positivo |
| **Amisoft** | Paquete aislado | Incidencias | Comprobado |
| **Amisoft** | Alto acoplamiento | Desarrollo de requerimientos | Comprobado |

| | | | |
|---|---|---|---|
| **Amisoft** | Baja Cohesión | Desarrollo de requerimientos | Comprobado |
| **Amisoft** | Baja Cohesión | Planificación | Comprobado |
| **Amisoft** | Baja Cohesión | Procesos Amisoft | Comprobado |
| **Amisoft** | Baja Cohesión | Solución técnica | Comprobado |
| **Amisoft** | Paquete aislado | Verificación | Comprobado |
| **Amisoft** | Inestabilidad | Medición y análisis | Comprobado |
| **Amisoft** | Inestabilidad | Desarrollo de requerimientos | Comprobado |
| **Small SPL** | Inestabilidad | Ingeniería de Dominio | Comprobado |

**Tabla 7. Errores potenciales encontrados en los modelos de proceso**

Después del análisis de los datos recogidos en el estudio de caso, se observa que los errores asociados a la inestabilidad, y el acoplamiento, no fueron concluyentes para la identificación de patrones de error. Por lo tanto los esfuerzos se centran en los problemas asociados a la cohesión, y se logra identificar un Patrón de error relacionado con la baja cohesión.

Para especificar este patrón, se realiza un análisis adicional para lograr determinar un umbral asociado con la cohesión, y con ello poder clasificar los paquetes de contenido de método que tengan una baja, media y alta cohesión, y de esta manera usar los colores semáforo, rojo, amarillo y verde, respectivamente para dibujar los nodos en el Blueprint. A continuación se muestran solo los problemas encontrados relacionados con el patrón de error Baja Cohesión, y así lograr identificar un patrón asociado con la cohesión.

**Amisoft*:*** en este modelo de proceso de software, un ingeniero de proceso con la ayuda de AVIMO-PS, se logró identificar 4 instancias del problema Baja Cohesión, que

corresponden a los nodos A, B, C, D, en la

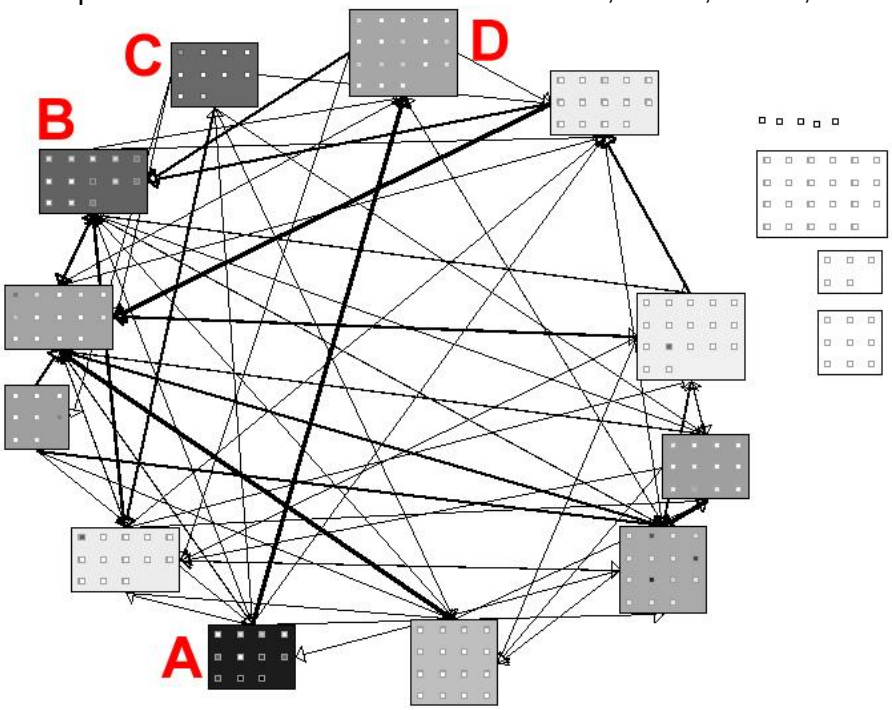



Figura 33. Estos nodos representan los paquetes de Desarrollo de requerimientos, Planificación, Procesos Amisoft y Solución Técnica.

**Tutelkan*:*** de manera similar, AVIMO-PS fue utilizado para encontrar 6 posibles anomalías asociadas con la Baja Cohesión, en los paquetes, Medición y Análisis, Planificación del proyecto, Pruebas, Gestión de Riesgos, Desarrollo de Requerimientos, Análisis y Diseño, los cuales se pueden ver en Figura 34, en los nodos A, B, C, D, E, F respectivamente.

**Rhiscom*:*** AVIMO-PS se utilizó para identificar y localizar dos posibles anomalías asociadas con la baja cohesión, en los paquetes de Requisitos y Comercial, los cuales son representados por los nodos A y B, respectivamente, como se observa en la

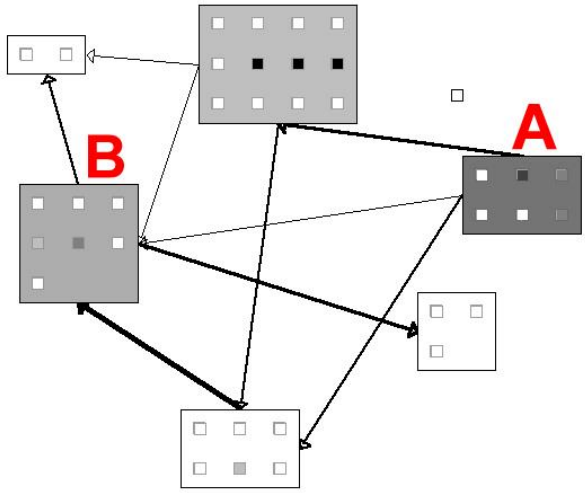



Figura 35.

**Small SPL:** En este modelo de proceso el sujeto de investigación no identificó algún problema asociado con la cohesión.

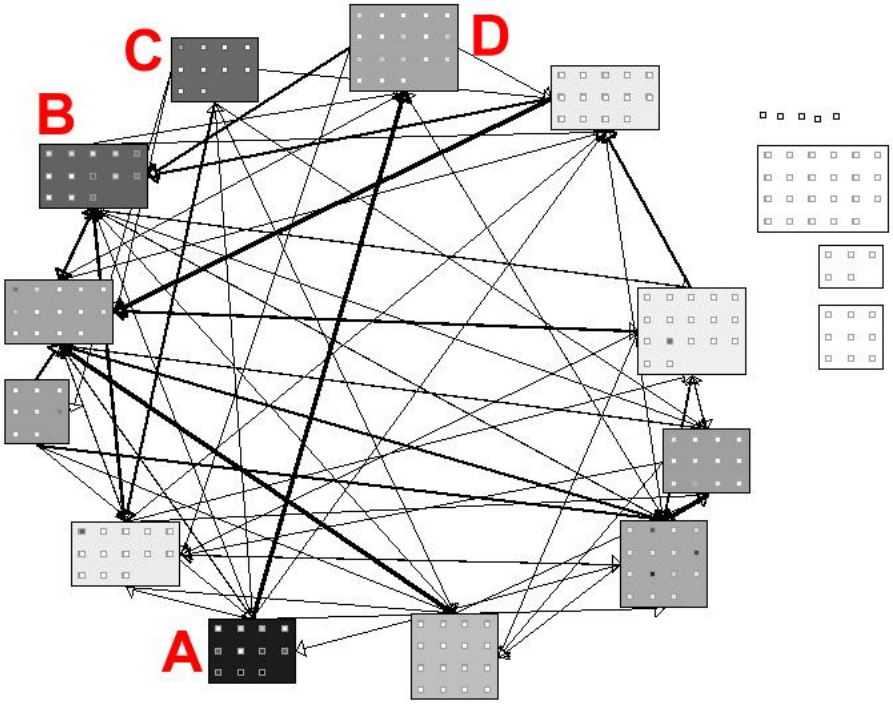



**Figura 33. Paquetes con Baja Cohesión en Amisoft**

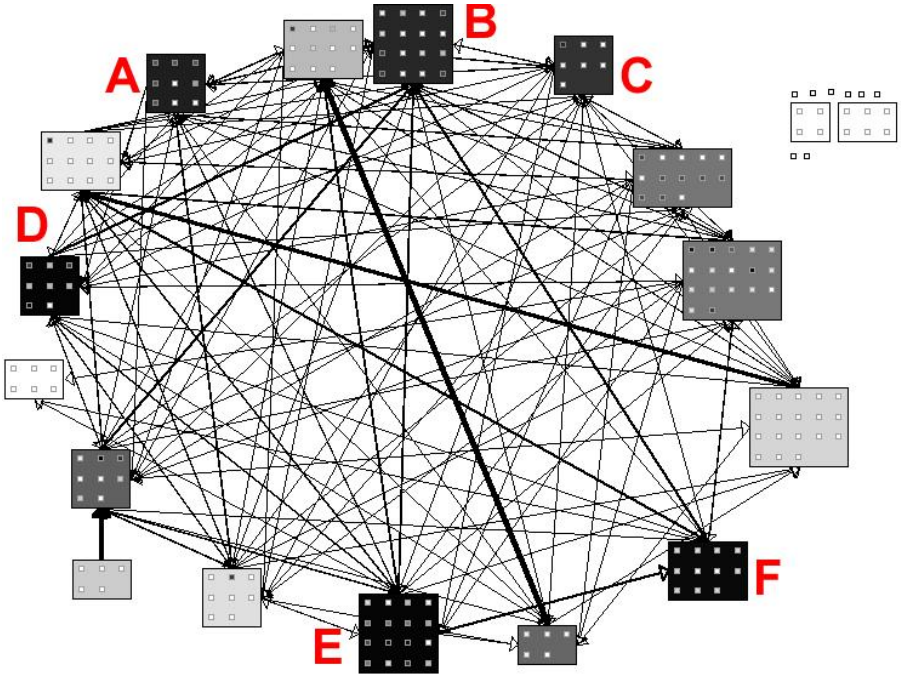



**Figura 34. Paquetes con Baja Cohesión en Tutelkan**

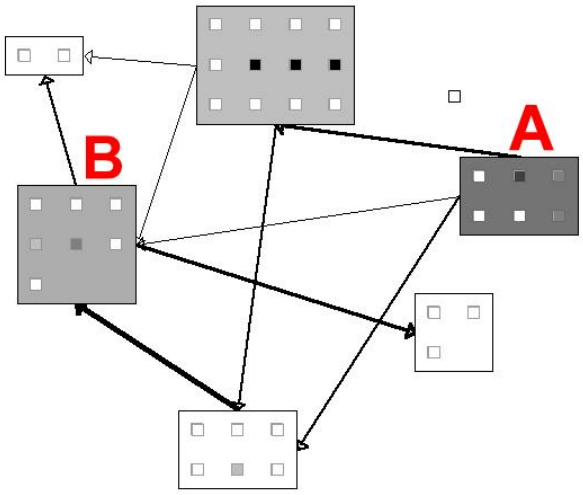



**Figura 35. Paquetes con Baja Cohesión en Rhiscom**

Todos los problemas confirmados en los modelos de proceso fueron notificados a los ingenieros de proceso responsables, con el fin de que ellos realicen el análisis y las adecuaciones pertinentes en la especificación que haya lugar, pero esto no se expone en este documento ya que no hace parte del alcance. Con el fin de establecer un umbral inferior y superior asociado a la cohesión, se utilizaron los datos recogidos durante el análisis de los modelos de proceso con AVIMO-PS. En la Tabla 8 se reflejan las anomalías en los paquetes de contenido de método, con una baja cohesión, al igual que el resultado obtenido de la **¡Error! No se encuentra el origen de la referencia.**, y el estado de la anomalía después de realizar su respectivo análisis.

| Modelo de proceso | Paquete de contenido de método | Cohesión | Estado |
|---|---|---|---|
| **Tutelkan** | Gestión de Riesgos | 0.13 | Comprobado |
| **Tutelkan** | Desarrollo de Requerimientos | 0.13 | Comprobado |
| **Amisoft** | Desarrollo de requerimientos | 0.18 | Comprobado |
| **Tutelkan** | Análisis y Diseño | 0.18 | Comprobado |
| **Tutelkan** | Medición y Análisis | 0.33 | Comprobado |
| **Tutelkan** | Planificación del proyecto | 0.38 | Comprobado |
| **Tutelkan** | Pruebas | 0.43 | Comprobado |
| **Amisoft** | Planificación del Proyecto | 0.46 | Comprobado |
| **Amisoft** | Solución técnica | 0.61 | Falso positivo |
| **Rishcom** | Comercial | 0.67 | Falso positivo |
| **Amisoft** | Procesos Amisoft | 0.7 | Falso positivo |
| **Rishcom** | Implementación | 0.71 | Falso |

| | | | |
|---|---|---|---|
| | | | positivo |

**Tabla 8. Anomalías de Baja Cohesión**

Después del análisis de los datos, y una vez validada la información de la Tabla 8. Anomalías de Baja Cohesión, puede servir como indicadores que llevan a identificar paquetes propensos a problemas de Baja Cohesión. Tales paquetes pueden identificarse para someterlos a inspecciones más intensas, o en algunos casos, rediseñarse.
Los umbrales se definen como “Los valores heurísticos que se usan determinar rangos de valores deseables y no deseables de métricas” [87]. Tomando este concepto, se establece un umbral de manera experimental, y así poder identificar los paquetes con *baja, media y alta cohesión*, para facilitar la identificación y localización las anomalías en el modelo de proceso de una manera más rápida.

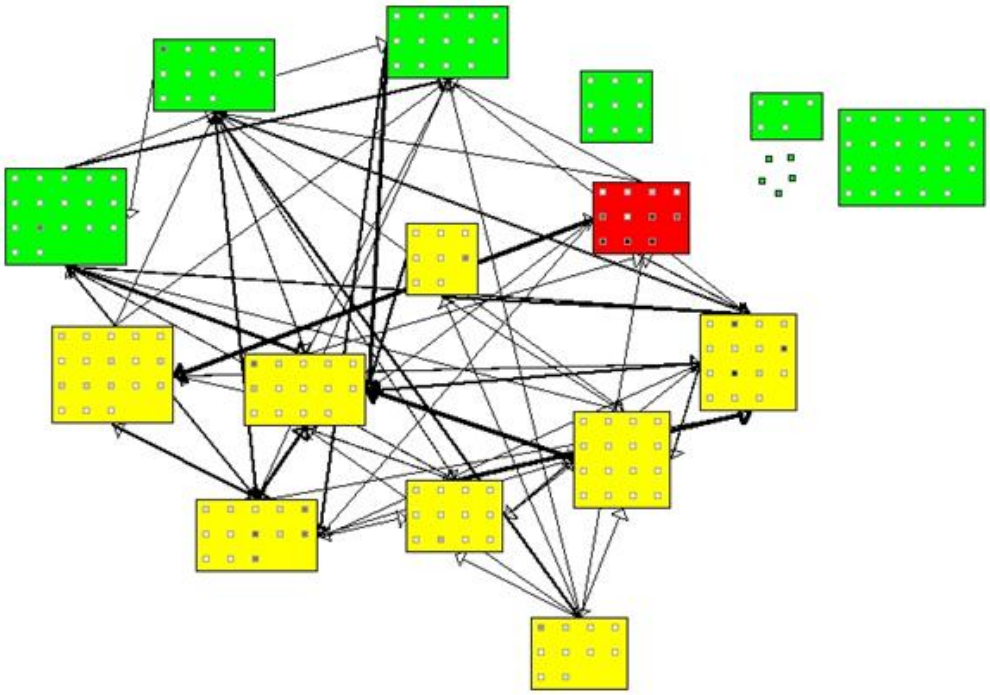
De esta manera, se organizan y separan los errores comprobados, y los que son falsos positivos, como se muestra en la Tabla 9. Con el fin de identificar un umbral, y con ello definir un Patrón de error.

| Comprobado | | | | | | | | Falso positivo | | | | Sin Problemas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0,13 | 0,13 | 0,18 | 0,18 | 0,33 | 0,38 | 0,43 | 0,46 | 0,61 | 0,67 | 0,7 | 0,71 | > |

**Tabla 9. Identificación del Umbral**

El patrón de baja cohesión, se determina por dos umbrales, que se logran identificar a partir de la Tabla 9, los cuales permiten identificar claramente tres regiones: Paquetes con problemas comprobados de baja cohesión, cuyos valores están entre 0 y 0.46 (color rojo); Paquetes cuya cohesión fue puesta en duda y por lo tanta se sometieron a análisis para confirmar o descartar el hallazgo, donde sus valores de cohesión son mayores a 0.46 y menores o iguales a 0.71 (color amarillo); y una tercera región, en donde están los paquetes que no se identificaron problemas asociados a la cohesión, y sus valores fueron mayores a 0.71.
El umbral superior (0.71) se determinó seleccionando el máximo valor de los falsos positivos, indicando que arriba de éste, ningún elemento fue cuestionado por su cohesión. El umbral inferior (0.46) corresponde al mayor valor de los problemas corroborados, es decir, debajo de éste no se encontró algún falso positivo, en todo paquete con un valor menor o igual a 0.46 se confirmó un problema de baja cohesión. Por lo tanto se establece un Patrón de baja cohesión, como se muestra en la Figura 35.


**Figura 36. Esquema de colores semáforo**



En la Figura 37. Patrón de Baja Cohesión en el modelo de proceso Amisoft se muestra, una visualización del modelo de proceso Amisoft, en el cual se ha aplicado el umbral de la cohesión, con el fin de crear el patrón de Baja Cohesión, y así destacar los paquetes que tienen baja, media y alta cohesión.

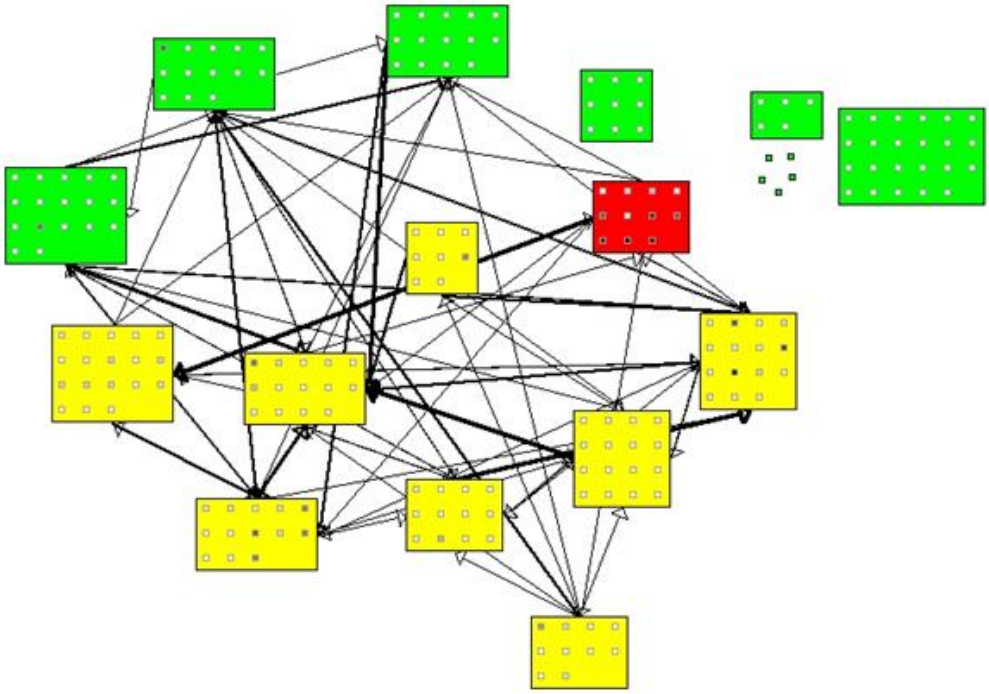


**Figura 37. Patrón de Baja Cohesión en el modelo de proceso Amisoft**

## 4.9 Amenazas de Validez

El resultado de los datos cuantitativos y cualitativos obtenidos en el estudio de caso, son difíciles de generalizar ya que al contar con poca disponibilidad de los sujetos investigados y de pocos modelos de procesos de software, no se logró realizar un estudio de caso con muchas unidades de análisis. Para mitigar ésta amenaza los modelos de procesos utilizados, se seleccionaron en una variedad de contextos con el fin de abarcar la mayor cantidad de posibilidades: de fuente abierta, fuente cerrada, basados en CMMI y los que no se basan en CMMI. También, los modelos utilizados abarcaron diferentes tamaños, 26(264), 7(83), 3 (181) y 20(312) paquetes de método (y número de elementos respectivamente). Los sujetos de investigación también fueron seleccionados para cubrir dos grados de experticia con el análisis visual. Es posible que los tiempos de verificación del modelo de proceso aumenten si se incrementa el tamaño y complejidad de los modelos de proceso, sin embargo, aunque el tiempo que tardaron los sujetos de investigación fue corto, la efectividad de los problemas hallados en los modelos fue bastante alta, lo que muestra las ventajas de AVIMO-PS.

## 4.10 Síntesis y Discusión

Para los ingenieros de proceso AVIMO-PS ha sido de gran utilidad para visualizar elementos importantes de un modelo de proceso elaborado. AVIMO-PS proporciona valiosa información que permite tener en cuenta el grado de modularidad de un modelo de proceso desde la definición del mismo, sin embargo es importante resaltar que un ingeniero de procesos que interactúe con AVIMO-PS debe tener claro los conceptos de acoplamiento, cohesión e inestabilidad, y por otro lado debe tener el conocimiento sobre las notaciones de las métricas, para interpretar los Blueprints generados. Un aspecto importante de los Blueprints generados por AVIMO-PS, es que éstos le ayudan al ingeniero de procesos a conocer el grado de acoplamiento del modelo de proceso, posicionando el cursor sobre un elemento automáticamente se marca en rojo los elementos que están asociados y de los cuales depende.

Para disminuir el esfuerzo utilizado para la interpretación de los Blueprints se requiere que el usuario tome una corta capacitación de los conceptos de acoplamiento, cohesión e inestabilidad con el fin de que sea más fácil la comprensión e identificación de los problemas o anomalías presentados en un modelo de proceso cargado en AVIMO-PS.

Teniendo en cuenta que de acuerdo al estudio de caso, en promedio se requiere de 1.75 horas para realizar el análisis de procesos con AVIMO-PS, resulta razonablemente viable para el grupo de ingeniería de procesos. El nivel de efectividad ha permitido detectar que los ingenieros de proceso han logrado identificar aspectos relevantes en los modelos de proceso evaluados. El indicador de la efectividad con un valor de 88,65%, nos dice que la AVIMO-PS cumple con su objetivo, sirviendo como herramienta al ingeniero de proceso para identificar por medio de los Blueprints, los defectos evidenciados en un modelo de proceso. Por otra parte, se obtiene que los ingenieros de procesos obtuvieran un nivel del comprensión del 87,5%, dando como resultado un aspecto que genera valor, teniendo en cuenta que la visualización de modelos de procesos es un factor novedoso dentro del área de la ingeniera de procesos, y se espera que al definir un patrón de error el nivel de efectividad sea incrementado, ya que facilita la identificación de paquetes problemáticos, reduciendo sustancialmente el conjunto de elementos a analizar.

Por otro lado es necesario resaltar que el prototipo implementado ha sido desarrollado en el lenguaje de programación Smalltalk, basándose en la plataforma de análisis de software MOOSE y el motor de visualización MONDRIAN, siendo un complemento a las herramientas de software de diseño de procesos actuales. Los Blueprints generados inicialmente(Ver
Figura 38)contenían mucha información la cual no era de mucha utilidad y por ende de compleja interpretación; estos Blueprints se fueron ajustando de tal forma que la información que se visualizara se interpretara de una mejor manera, disminuyendo como tal el esfuerzo por parte de los ingenieros de procesos al momento de identificar los errores en los modelos de procesos.

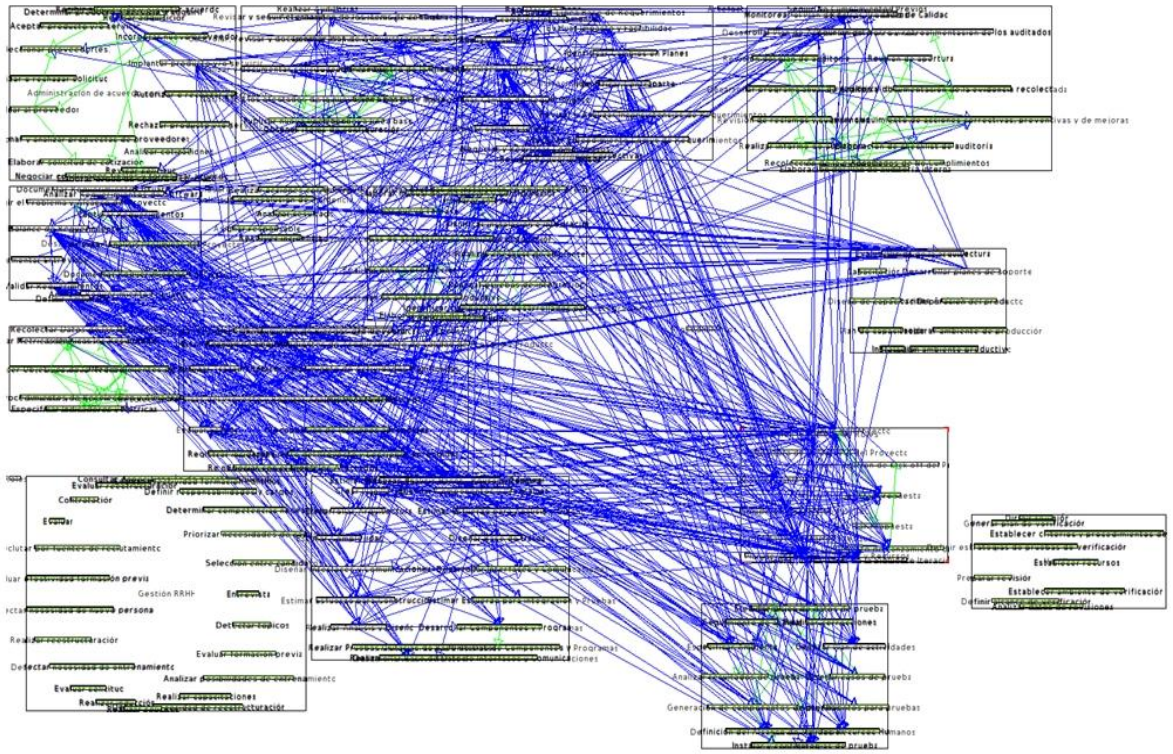


**Figura 38. Blueprint inicial**

Basados en los resultados del estudio de caso, los ingenieros de procesos reportan que el nivel de complejidad es relativamente bajo considerando como referencia la experiencia con el manejo de EPFC y otras herramientas de modelado, y que la eficiencia se puede considerar alta, ya que de acuerdo a los resultados y con una corta capacitación, se logró en un tiempo corto identificar un gran número de problemas. Por otro lado, el contar con ingenieros de procesos fue de gran utilidad al momento de interactuar con la herramienta. La impresión de los ingenieros de proceso en el momento de realizar el análisis de los modelos fue aceptable, y podemos concluir que no se generó frustración al utilizar la herramienta para el análisis visual de modelos de procesos de software.

# 5 Conclusiones, Limitaciones y Trabajo Futuro

Los modelos de proceso software de manera natural evolucionan, es decir, se adaptan al proyecto y en el tiempo, y además este es potencialmente reutilizable. Es importante que los modelos de proceso estén definidos para el cambio y/o una posible evolución. Dicha importancia de los procesos en las empresas desarrolladoras de software se generan desde iniciativas de mejoramiento, dado que los procesos de software deben modificarse y evolucionar sistemáticamente guiándose por nuevos objetivos organizacionales para adaptarse y satisfacer los nuevos escenarios y su entorno tan cambiante. Por ello es elemental conocer los errores que se pueden presentar en la definición de un modelo de proceso, sin embargo realizar esta tarea resulta un poco complicado, ya que no se cuenta con herramientas intuitivas, o visuales para soportarlo.

En la actualidad, las métricas existentes para evaluar los modelos de procesos de software están en un área con poca investigación. En este trabajo, adaptamos varias métricas de acoplamiento y cohesión de la ingeniería de software al dominio de procesos de software. Nuestras adaptaciones se basan en las similitudes entre los conceptos de los elementos de SPEM2.0 y POO. Además, representaban las relaciones que existen entre las elementos de contenido de método: tareas, roles y artefactos. Además, las métricas de acoplamiento y cohesión propuestas cumplen varias propiedades que las hacen adecuadas para la identificación de problemas en un modelo de proceso de software.

## 5.1 Conclusiones

El poder de la visualización ha venido siendo utilizado para evaluar aspectos del modelo de proceso de software. Este es el caso de AVISPA que evalúa la correctitud de los modelos de proceso y en este trabajo de grado, AVIMO-PS evalúa aspectos de modularidad. AVIMO-PS es un prototipo software que explota el poder de la visualización de los modelos de procesos de software, ayudando a entender de una mejor manera la distribución de los elementos de contenido de método, ofreciendo un enfoque que no sólo ayuda a diseñar modelos de proceso de software, sino que también ayuda a determinar la inestabilidad, y el acoplamiento y cohesión de los paquetes de contenido de método. Esto muestra que el camino de la visualización de modelos de proceso apenas comienza y que éste puede abarcar más factores problemáticos del modelo de proceso, por ejemplo la usabilidad, el desempeño y otros atributos de calidad.
Mediante el estudio de caso de este trabajo de grado se ha podido evaluar de manera objetiva la utilidad y facilidad de manejo del prototipo AVIMO-PS. Trabajar con los estudios de caso, requiere definir meticulosamente criterios que permitan realizar esta labor de manera integral y coherente. Que en nuestro caso nos permitió, evidenciar que AVIMO-PS es una herramienta útil para al análisis visual de la modularidad de un modelo

de procesos de software y gracias a ello, es posible mejorar los aspectos de modificabilidad, aportando en su mantenibilidad y soporte para evoluciones futuras.

El prototipo AVIMO-PS facilita la tarea de evaluación de los modelos de proceso de software, y la identificación visual de problemas en la modularidad, que tienen impacto directo sobre la modificabilidad del modelo de proceso durante su evolución. Esto permite a los ingenieros de procesos detectar problemas y analizar datos en modelos definidos de manera temprana, utilizando la capacidad humana para visualizar e interpretar las vistas de los modelos de procesos, logrando evaluar la coherencia y confiabilidad de un proceso de software definido estáticamente en una forma rápida, justo antes de ponerlos a prueba en proyectos reales. Lo que lo convierte en una posible herramienta para organizaciones que trabajan en la industria del software, quienes enfocan sus esfuerzos en la definición de procesos de software teniendo en cuenta las particularidades de cada proyecto u organización.
Debido a que la visualización de procesos de software se utiliza poco en nuestra industria, los resultados de este trabajo aportarán un grano de arena para optimizar el diseño y elaboración de procesos de software con el fin de reducir los costos, mejorar la productividad, la gestión y el manejo del tiempo.

Los indicadores planteados en el estudio de caso ayudaron a ponderar de una forma más objetiva el resultado del mismo, logrando medir la efectividad de la herramienta, complejidad y comprensión percibida por el usuario, y la facilidad de uso que provee la herramienta, lo que nos ayudó a identificar aspectos a mejorar en la herramienta.

## 5.2 Limitaciones

Es importante anotar que cómo todo trabajo de investigación, éste también tiene limitaciones. El prototipo AVIMO-PS está dirigido solo a los modelos de procesos de software especificados formalmente en SPEM2.0, esto puede ser uno de sus principales limitaciones ya que es difícil y costoso definir formalmente un proceso completo, principalmente para las pequeñas organizaciones de software, y además existen otros lenguajes y herramientas para su especificación. Particularmente, las especificaciones XML de un proceso SPEM2.0, deben de contener paquetes de contenido de método, ya que el prototipo AVIMO-PS, permite generar los Blueprint centrando su procesamiento en los paquetes de contenido de método, resaltando sus ventajas. Sin embargo si se evalúa un proceso de software que no tiene especificados los paquetes de contenido de método, afectarían los resultados aquí expresados. Además, la cohesión y acoplamiento en AVIMO–PS sólo se mide a nivel de paquetes de contenido de método, hay que considerar que en SPEM2.0 además existen paquetes de proceso, plugins y configuraciones, a través de los cuales se podrían hacer análisis similares. Los patrones han surgido del análisis de los resultados de la aplicación de Blueprints, teóricamente tienen una efectividad del 100%, sin embargo ésta debe someterse a una

validación exhaustiva en nuevos estudios de caso y experimentos para determinar su nueva efectividad.

## 5.3 Lecciones Aprendidas y Problemas Enfrentados

La gestión del riesgo técnico es un factor fundamental en el momento de realizar un trabajo de grado, en nuestro caso el investigar y trabajar sobre las tecnologías y herramientas de las que podíamos soportarnos en paralelo con la formulación del anteproyecto, nos permitió sentar unas bases sólidas para analizar el alcance del proyecto y adelantar trabajo para terminarlo dentro de los plazos establecidos. En la práctica, un caso de estudio requiere un diseño muy detallado y un análisis de los posibles inconvenientes que pueden surgir en el momento de realizarlo, de tal manera que si estos problemas potenciales lleguen a suceder, se los pueda enfrentar de una manera proactiva.

Moose, es una plataforma de software, que ofrece múltiples servicios que van desde la importación y el análisis de datos, el modelado, la medición, la consulta, la minería y la construcción de herramientas de análisis interactivos y visuales. Sin embargo esta plataforma es relativamente nueva, y por ello no cuenta con la documentación suficiente o deseada, para facilitar un rápido aprendizaje, y obtener un máximo provecho de todo su potencial. Así, proyectos futuros deben considerar estrategias para manejar la curva lenta de aprendizaje de la herramienta.

## 5.4 Trabajo Futuro

Basados en la limitaciones y lecciones aprendidas del proyecto, en un futuro, el grupo IDIS de la Universidad del Cauca pretende continuar trabajando con AVISPA y AVIMO-PS. existen diferentes extensiones posibles a este trabajo de grado, una de ellas es continuar con la clasificación, mejora y ampliación de las métricas propuestas, con el fin de incluir conceptos importantes de medición adicionales tales como la usabilidad, la complejidad y la cohesión en los elementos, entre otros atributos de calidad de los modelos de procesos de software.

Además, realizar un análisis sobre la importancia relativa de las métricas propuestas, con el fin de poder priorizar su uso, con la ayuda de futuras propuestas de modularidad de modelos de software.
Parte del trabajo futuro técnico corresponde con mejorar la herramienta AVISPA y el prototipo AVIMO-PS, de tal forma que tengan la capacidad de leer, analizar y procesar la información de otros elementos de contenido de método de SPEM2.0, tales como guías de trabajo, steps, etc., y así, ampliar el metamodelo y la capacidad de análisis, contribuyendo al entendimiento de los modelos de procesos de software importados a

AVIMO-PS, lo cual permite la implementación de nuevos Blueprints, y la identificación de nuevos patrones de errores, mejorando la capacidad de análisis de modelos de procesos.

Por otro lado, es necesario realizar trabajo experimental, con diseños y ejecuciones de nuevos estudios de caso, para los cuales se deberá determinar el tipo de estudio, los sujetos de investigación, además de diseñar indicadores, métricas, e instrumentos. En cuanto a las unidades de análisis, estas también serán los modelos de procesos de software especificados en SPEM 2.0, tanto de código abierto, como de código privativo o industriales, con el fin de abordar procesos de diferentes tamaños, propósitos, complejidades y contextos. Y así continuar validando las métricas, los Blueprints y el patrón de error propuesto en este trabajo de grado, para lograr un producto industrial más completo. Una vez realizados estos nuevos estudios de caso, se tendrá una gran variedad de datos obtenidos empíricamente, lo cual ayudará a determinar los umbrales del patrón de error, en una forma mucho más confiable. Y así mismo será posible identificar otros patrones de error asociados a la modularidad de los modelos de procesos.

Gracias a los resultados obtenidos a través de las encuestas realizadas a los sujetos de investigación, se obtuvo que la herramienta AVIMO-PS tiene aspectos a mejorar, tanto de comprensibilidad, como de usabilidad. Ya que un Blueprint de un modelo de proceso muy grande y extenso puede resultar complejo para un ingeniero de proceso, puede ser importante trabajar en la mejora de las visualizaciones, con el fin de mejorar la legibilidad, navegabilidad y contribuir en mejorar la comprensión de los Blueprint. Además es importante en trabajar en aspectos de usabilidad, tales como facilidad de uso, eficiencia, intituividad, memorabilidad, características que mejoren la experiencia del usuario y que aumente su satisfacción.

# 6 REFERENCIAS BIBLIOGRÁFICAS


[1] J. Alegría*, et al.*, "Software Process Model Blueprints," *New Modeling Concepts for Today's Software Processes,* pp. 273-284, 2010.

[2] J. A. Calvo-Manzano*, et al.*, "Perfiles del ciclo de vida del software para pequeñas empresas: los informes técnicos ISO/IEC 29110," *Revista Española de Innovación, Calidad e Ingeniería del Software,* vol. 4, 2008.

[3] F. Pino*, et al.*, "Modelo para la Implementación de Mejora de Procesos en Pequeñas Organizaciones Software," *XII Jornadas de Ingeniería del Software y Bases de Datos, JISBD,* pp. 326–335, 2007.

[4] M. Qasaimeh*, et al.*, "Comparing Agile Software Processes Based on the Software Development Project Requirements," in *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*, 2008, pp. 49-54.

[5] F. Ruiz, "MANTIS: Definición de un Entorno para la Gestión del Mantenimiento de Software," *Departamento de Informática,* 2003.

[6] D. Poshyvanyk and A. Marcus, "The conceptual coupling metrics for object-oriented systems," in *Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on*, 2006, pp. 469-478.

[7] T. M. Shaft and I. Vessey, "The role of cognitive fit in the relationship between software comprehension and modification," *MIS Quarterly,* vol. 30, pp. 29-55, 2006.

[8] J. Garzás*, et al.*, "Una aplicación de ISO/IEC 15504 para la evaluación por niveles de madurez de PYMEs y pequeños equipos de desarrollo," *Revista Española De Innovación, Calidad e Ingeniería Del Software (REICIS),* vol. 5, pp. 88-98, 2009.

[9] J. Diaz*, et al.*, "Mapping CMMI level 2 to Scrum practices: An experience report," in *Software Process Improvement*, ed: Springer, 2009, pp. 93-104.

[10] V. Gruhn, *Validation and verification of software process models*: Springer, 1991.

[11] J. Mendling*, et al.*, "Thresholds for error probability measures of business process models," *Journal of Systems and Software,* vol. 85, pp. 1188-1197, 2012.

[12] J. Ge*, et al.*, "Modeling multi-view software process with object petri nets," in *Software Engineering Advances, International Conference on*, 2006, pp. 41-41.

[13] J. A. Hurtado Alegría*, et al.*, "Analyzing software process models with AVISPA," in *Proceedings of the 2011 International Conference on Software and Systems Process*, 2011, pp. 23-32.

[14] J. A. H. Alegrıa*, et al.*, "AVISPA: Localizing Improvement Opportunities in Software Process Models," 2010.

[15] B. J. Williams and J. C. Carver, "Characterizing software architecture changes: A systematic review," *Information and Software Technology,* vol. 52, pp. 31-51, 2010.

[16] A. April and C. Laporte, "An Overview of Software Engineering Process and Its Improvement," 2009.

[17] R. Kaur and J. Sengupta, "New Approach in Software Development-Fusion Process Model," *Journal of Software Engineering and Applications,* vol. 3, pp. 998-1004, 2010.

[18] P. O. Bengtsson*, et al.*, "Analyzing software architectures for modifiability," *Journal of Systems and Software,* 2000.

[19] S. F. Ahmad*, et al.*, "A Comparative Study of Software Quality Models," *International Journal of Science, Engineering and Technology Research,* vol. 2, pp. pp: 172-176, 2013.

[20] L. J. Osterweil, "Software processes are software too, revisited: an invited talk on the most influential paper of ICSE 9," 1997.
[21] P. Kruchten*, et al.*, "The decision view's role in software architecture practice," *Software, IEEE,* vol. 26, pp. 36-42, 2009.
[22] J. Montilva, "Aplicando modelos de procesos de software al desarrollo de aplicaciones hipermedia," 1996, pp. 870-881.
[23] J. Al Dallal, "A design-based cohesion metric for object-oriented classes," *International Journal of Computer Science and Engineering,* vol. 1, pp. 195-200, 2007.
[24] T. M. Meyers and D. Binkley, "An empirical study of slice-based cohesion and coupling metrics," *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 17, p. 2, 2007.
[25] S. T. Acuña*, et al.*, "The software process: Modelling, evaluation and improvement," *Handbook of Software Engineering and Knowledge Engineering,* vol. 1, pp. 193-237, 2001.
[26] IDIS. (2013). *Investigación y Desarrollo en Ingeniería del Software*. Available: http://www.unicauca.edu.co/idis/
[27] A. Fuggetta and A. L. Wolf, *Software process*: John Wiley & Sons, Inc., 1996.
[28] E. Rolón*, et al.*, "Aplicación de métricas software en la evaluación de modelos de procesos de negocio," *Revista Electrónica de la Sociedad Chilena de Ciencia de la Computación,* vol. 6, 2005.
[29] R. McLeod and G. Schell, *Management Information Systems 8/e*: Prentice-Hall, Inc, 2001.
[30] G. Canfora and F. Ruiz González, "Procesos Software: características, tecnología y entornos," *Novática: Revista de la Asociación de Técnicos de Informática,* pp. 5-8, 2004.
[31] T. Samaniefo and M. Zulay, "Estudio comparativo entre los estandares ISO/IEC TR 15504 y CMMI," 2007.
[32] R. Pressman, " Software Engineering: A Practitioner's Approach, McGraw-Hill Book Company," 2005.
[33] J. Guerrero and P. Santiago, "Propuesta de un modelo de evaluación y mejora de los procesos de ingeniería en el desarrollo de software para la empresa Icono Sistemas," LATACUNGA/ESPE/2012, 2012.
[34] A. Finkelstein, Kramer J. ,Cheng, B. ,de Lemos, R., *, et al.*, "Software engineering for self-adaptive systems: A research roadmap," *Software Engineering for Self-Adaptive Systems,* pp. 1-26, 2009.
[35] I. Sommerville, *Ingeniería del software*: Pearson Educación, 2005.
[36] S. T. Acuña*, et al.*, "A culture-centered multilevel software process cycle model," 2000, pp. 775-775.
[37] S. Acuña*, et al.*, "Software engineering and knowledge engineering software process: formalizing the who's who," 2000, pp. 221-230.
[38] K. S. Devesh*, et al.*, "Square Model-A Proposed Software Process Model for BPO based Software Applications," *International Journal of Computer Applications,* vol. 13, pp. 33-36, 2011.
[39] M. R. Carreira*, et al.*, "Mejora de los procesos software utilizando simulación e integración de técnicas," 2001.
[40] S. T. Acuña*, et al.*, "The software process: Modelling, evaluation and improvement," *Handbook of Software Engineering and Knowledge Engineering,* pp. 193–237, 2001.
[41] M. García*, et al.*, "Implantación de las normas ISO/IEC 15504 e ISO/IEC 12207 con métodos ágiles y SCRUM," *Agil Spain,* 2010.

[42] F. Pino*, et al.*, "Adaptación de las normas ISO/IEC 12207: 2002 e ISO/IEC 15504: 2003 para la evaluación de la madurez de procesos software en países en desarrollo," *IEEE Latin America Transactions,* vol. 4, pp. 17-24, 2006.
[43] D. Hoyle, *ISO 9000 Quality Systems Handbook-updated for the ISO 9001: 2008 standard*: Routledge, 2012.
[44] T. P. Rout, "ISO/IEC 15504—Evolution to an international standard," *Software Process: Improvement and Practice,* vol. 8, pp. 27-40, 2004.
[45] S. U. Team, "Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document," 2011.
[46] P. COMPETISOFT, "COMPETISOFT-Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica. Versión 0.2," ed: Diciembre, 2006.
[47] H. Oktaba*, et al.*, "Método de Evaluación de procesos para la industria del software EvalProSoft," ed: Versión, 2004.
[48] OMG, "Software Process Especification Meta-model Specification.," *OMG Object Management Group, 2.0 final adopted edition,* 2007.
[49] M. Lanza and S. Ducasse, "Polymetric views-a lightweight visual approach to reverse engineering," *Software Engineering, IEEE Transactions on,* vol. 29, pp. 782-795, 2003.
[50] M. M. Lehman*, et al.*, "Implications of evolution metrics on software maintenance," in *Software Maintenance, 1998. Proceedings. International Conference on*, 1998, pp. 208-217.
[51] F. Brito e Abreu and M. Goulão, "Coupling and Cohesion as Modularization Drivers: Are we being over-persuaded?," 2001, pp. 47-57.
[52] H. Abdeen*, et al.*, "Modularization Metrics: Assessing Package Organization in Legacy Large Object-Oriented Software," in *Reverse Engineering (WCRE), 2011 18th Working Conference on*, 2011, pp. 394-398.
[53] S. G. Eick*, et al.*, "Does code decay? assessing the evidence from change management data," *Software Engineering, IEEE Transactions on,* vol. 27, pp. 1-12, 2001.
[54] E. B. Allen*, et al.*, "Measuring coupling and cohesion of software modules: an information-theory approach," in *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, 2001, pp. 124-134.
[55] I. Gorton and L. Zhu, "Tool support for just-in-time architecture reconstruction and evaluation: an experience report," in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, 2005, pp. 514-523.
[56] R. Martin, "OO design quality metrics," *An analysis of dependencies,* 1994.
[57] A. Fuggetta, "Software process: a roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 25-34.
[58] R. Harrison*, et al.*, "Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems," *Journal of Systems and Software,* vol. 52, pp. 173-179, 2000.
[59] L. C. Briand*, et al.*, "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs," *Software Engineering, IEEE Transactions on,* vol. 27, pp. 513-530, 2001.
[60] M. Marchesi, "OOA metrics for the Unified Modeling Language," in *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, 2003, pp. 67-73.
[61] M. Saeki, "Embedding metrics into information systems development methods: An application of method engineering technique," in *Advanced information systems engineering*, 2003, pp. 374-389.

[62] M. Genero*, et al.*, "Building UML class diagram maintainability prediction models based on early metrics," in *Software Metrics Symposium, 2003. Proceedings. Ninth International*, 2003, pp. 263-275.
[63] D. Miranda*, et al.*, "Empirical validation of metrics for UML statechart diagrams," in *Enterprise Information Systems V*, ed: Springer, 2005, pp. 101-108.
[64] G. Canfora*, et al.*, "A family of experiments to validate metrics for software process models," *Journal of Systems and Software,* vol. 77, pp. 113-129, 2005.
[65] Michael D. Lee*, et al.*, " An Empirical Evaluation of Cherno_ Faces, Star Glyphs, and Spatial Visualizations for Binary Data.
," *Australian Computer Society, Inc,* pp. 1-10, 2003.
[66] S. Christov*, et al.*, "Using Event Streams to Validate Process Definitions," *University of Massachusetts, Amherst,* 2009.
[67] K. El Emam, "A methodology for validating software product metrics," 2010.
[68] M. A. Babar, "A framework for supporting the software architecture evaluation process in global software development," *Software Engineering, IEEE Transactions on,* pp. 93-102, 2009.
[69] M. Lungu and M. Lanza, "Softwarenaut: Exploring hierarchical system decompositions," 2006, pp. 2 pp.-354.
[70] A. Brühlmann*, et al.*, "Enriching reverse engineering with annotations," *Model Driven Engineering Languages and Systems,* pp. 660-674, 2010.
[71] J. A. Duraes and H. S. Madeira, "Emulation of software faults: A field data study and a practical approach," *IEEE Transactions on Software Engineering,* pp. 849-867, 2006.
[72] B. Livshits and T. Zimmermann, "DynaMine: finding common error patterns by mining software revision histories," *ACM SIGSOFT Software Engineering Notes,* vol. 30, pp. 296-305, 2005.
[73] R. Wettel*, et al.*, "Software systems as cities: A controlled experiment," 2011.
[74] F. Perin, "MooseJEE: A moose extension to enable the assessment of JEAs," 2010, pp. 1-4.
[75] J. A. Hurtado Alegría*, et al.*, "Avispa: a tool for analyzing software process models," *Journal of Software: Evolution and Process,* 2013.
[76] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM,* vol. 15, pp. 1053-1058, 1972.
[77] B. Meyer, *Object-oriented software construction* vol. 2: Prentice hall New York, 1988.
[78] A. J. Riel, *Object-oriented design heuristics*: Addison-Wesley Longman Publishing Co., Inc., 1996.
[79] E. Yourdon and L. L. Constantine, *Structured design: fundamentals of a discipline of computer program and systems design*: Prentice-Hall, Inc., 1979.
[80] T. J. McCabe, "A complexity measure," *Software Engineering, IEEE Transactions on,* pp. 308-320, 1976.
[81] R. C. Martin, *Agile software development: principles, patterns, and practices*: Prentice Hall PTR, 2003.
[82] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering,* vol. 14, pp. 131-164, 2009.
[83] R. K. Yin, *Case study research: Design and methods* vol. 5: SAGE Publications, Incorporated, 2008.
[84] SUMI. (2013). *Software Usability Measurement Inventory*. Available: http://www.ucc.ie/hfrg/questionnaires/sumi/index.html

[85] J. A. Certuche*, et al.*, "Técnicas de usabilidad y accesibilidad orientadas a procesos de desarrollo de softwar," 2009.
[86] K. E. Wiegers, *Peer reviews in software: A practical guide*: Addison-Wesley Boston, 2002.
[87] K. El Emam*, et al.*, "The confounding effect of class size on the validity of object-oriented metrics," *Software Engineering, IEEE Transactions on,* vol. 27, pp. 630-650, 2001.