UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ELECCIÓN ENTRE PROCESOS AUTOMÁTICAMENTE ADAPTADOS Y PROCESOS PREDEFINIDOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

FELIPE IGNACIO GONZÁLEZ MARTÍNEZ

PROFESOR GUÍA:
MARÍA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
ALEXANDRE BERGEL
ROMAIN ROBBES
MARCOS SEPÚLVEDA FERNÁNDEZ

Este trabajo ha sido parcialmente financiado por FONDEF (Chile), Proyecto ADAPTE, número: D09I1171 y FONDEF IDeA, CONICYT, Proyecto GEMS, número: IT13I20010.

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE: Ingeniero Civil en Computación y
grado de Magíster en Ciencias, Mención Computación.
POR: Felipe Ignacio González Martínez
FECHA: 19/06/2015
PROF. GUÍA: María Cecilia Bastarrica Piñeyro

ELECCIÓN ENTRE PROCESOS AUTOMÁTICAMENTE ADAPTADOS Y PROCESOS PREDEFINIDOS

Durante los últimos años en Chile ha proliferado el uso de procesos en las empresas de software, asegurando normas de calidad para los clientes mientras permite acotar los costos de producción y los plazos de entrega para la empresa. Definir un proceso es una tarea intensiva en conocimientos y en tiempo, que además necesita de guía experta para una correcta ejecución. Más aún cuando no existe un proceso de software que sea adecuado en todas las situaciones.

Muchas empresas utilizan un conjunto de procesos predefinidos para distintos escenarios, que corresponden a los tipos de proyectos más comunes. En cambio ADAPTE, un proyecto Fondef, propone una alternativa de adaptación basada en *Model Driven Engineering* que obtiene procesos específicos de forma automática, que requiere definir un proceso general y el contexto para cada proyecto. Esta tesis propone evaluar si esta adaptación automática posee algún beneficio, con respecto a la estrategia utilizada previamente por las empresas, es decir, un conjunto de procesos previamente definidos.

Para comparar las estrategias es necesario considerar que tanto los procesos predefinidos como la adaptación automática utilizan como entrada el contexto del proyecto. Se ha definido un conjunto de contextos para comparar de forma correspondiente los procesos adaptados de cada estrategia. Los procesos se evaluarán utilizando una comparación de modelos, donde el beneficio es medido en la cantidad de tareas extras y tareas faltantes. Asimismo, se desea utilizar un *merge* entre los procesos que sean más similares para refinar el resultado de los procesos predefinidos.

La validación involucró a dos empresas chilenas, Rhiscom y Mobius, que han implantado la adaptación automática y que anteriormente utilizaban un conjunto de procesos predefinidos. Al considerar que la adaptación automática genera procesos óptimos para cada contexto, fue posible evaluar que incluso variaciones pequeñas en un contexto repercuten en la productividad y calidad del proceso predefinido seleccionado. Ahora bien, el *merge* entre dos procesos predefinidos permite disminuir la cantidad de tareas faltantes, pero aumentando considerablemente las tareas extras. Esto significa una mejora en la calidad del proceso, pero a costa de su productividad.

A mi familia.

Agradecimientos

Gracias a los miembros de la comisión, por su tiempo y disposición para evaluar mi trabajo. Especialmente a la profesora Cecilia, que sin su guía no hubiera dado este paso.

Gracias a todos los miembros de los proyectos ADAPTE y GEMS.

Gracias a mis amigas y amigos, que han sido lo más importante en estos años. Los cafés, almuerzos y cervezas es decir poco. Muchas gracias por los recuerdos para cuando seamos viejos.

Gracias a mi madre, mi padre y mis hermanas. Gracias por las lecciones de vida y los afectos, por enseñarme a caminar, a contar, a leer, a jugar a ser viejitos, a jugar pool, a carretear en la playa, a tener carácter y sentido del humor. Todo lo que soy, con todos los matices, se los debo a ustedes.

Tabla de Contenido

Índice de tablas

Índice de ilustraciones

Capítulo 1

Introducción

Actualmente en Chile coexisten tanto organizaciones multinacionales como empresas locales, que captan los requerimientos y necesidades en el rubro de las tecnologías de información. Es por esto, que durante los últimos años la industria nacional ha reconocido a las pequeñas y medianas empresas de software por contribuir de manera sostenida a la economía local mediante la explotación de nichos de negocio específicos.

Las pequeñas empresas de software, o *Small Software Enterprises* (SSE), se caracterizan por poseer no más de 50 empleados y de efectuar un conjunto de actividades recurrentes, e.g., implementación de software, consultoría de software, integración de soluciones y desarrollo de software a medida; además de contar con una facturación no mayor a 10.8 millones de dólares [24].

Por otro lado, la encuesta del año 2008 realizada por la Asociación Gremial de las empresas chilenas desarrolladoras de software (GECHS) [3], indica que el desarrollo de software a medida es lo que más contribuye en la facturación dentro de las empresas chilenas. Según el mismo estudio se espera un crecimiento en la exportación del servicio debido a la subcontratación de procesos de negocios de un país a otro, efecto conocido como *offshoring*. Otro factor a recalcar es el creciente interés en las certificaciones CMM/CMMI e ISO 9001, que han probado mejorar la imagen y la administración dentro de estas empresas y por consiguiente la satisfacción del cliente.

Implementar cualquiera de estos estándares requiere de la guía de un usuario experto, que es una inversión muy grande para una SSE. Con el objetivo de abordar este problema, ADAPTE, proyecto Fondef financiado por Conicyt Chile, ha implementado y aplicado en SSE chilenas una estrategia de formalización y adaptación de procesos o *process tailoring*. Ésto plantea adaptar automáticamente un proceso general al escenario o contexto del proyecto para obtener un proceso a medida para el nuevo proyecto. Se llamará a esto adaptación automática de procesos.

Si bien la adaptación automática de procesos ha sido aplicada en diferentes SSE chile-

nas [38, 40], confirmando su factibilidad técnica [39] y usabilidad [10]. Cabe indicar que la estrategia, por como ha sido construída, otorga un proceso que contiene las tareas necesarias y ninguna extra. Sin embargo, no se ha evaluado su costo efectividad con respecto a la estrategia utilizada anteriormente por varias de las empresas: un conjunto de procesos basados en plantillas o procesos predefinidos.

1.1. Oportunidad

Una empresa que no haya adoptado la estrategia propuesta por ADAPTE enfrenta una serie de desafíos al planificar un nuevo proyecto. Por ejemplo, la empresa tiene la posibilidad de aplicar un único proceso de software sin adaptar al contexto, o bien puede definir el proceso a utilizar desde cero cada vez. Estas alternativas tienen una alta probabilidad de generar resultados diferentes en escenarios similares, los cuáles no son sistemáticos ni comparables. Por otro lado, la empresa puede decidir no utilizar proceso alguno en sus proyectos, lo que puede transformarse en costos y riesgos no programados.

Formalizar un proceso de software contempla definir, bajo estándares formales, las especificaciones del proceso de desarrollo de una empresa, además de su evolución [26]. Por ejemplo, el costo de adoptar el modelo CMMI alcanza un valor de 25,000 dólares que contempla entrenamiento, evaluación externa y los costos de registro [44]; mientras que la cantidad de horas-persona que deben ser invertidas para definir un proceso, en experiencia del autor, puede alcanzar a más de 100 horas y como toda tarea humana, es propensa a errores. Incluso al considerar los costos iniciales, formalizar los procesos de software para una empresa significa no sólo un incremento en su capacidad de administración y un fortalecimiento de las relaciones comerciales con sus clientes, sino también una mejora en el cumplimiento de plazos, costos y la calidad del producto entregado.

Si bien los beneficios de formalizar son claros, no se ha cuantificado la mejora de aplicar la propuesta del proyecto ADAPTE. Esta tesis busca establecer un análisis de costo efectividad al comparar los resultados, de forma sistemática, entre la adaptación automática y un conjunto de procesos predefinidos. Se elige esta última estrategia por haber sido utilizada por las empresas anteriormente y por ser repetible dentro de ellas. El resultado de esta tesis permitirá a las empresas tomar una decisión informada sobre el impacto de adoptar la estrategia de adaptación automática propuesta por ADAPTE. De esta manera, se busca no sólo fortalecer los impactos de la calidad de los productos, sino también en los procesos utilizados, estableciendo un paso concreto en la profesionalización de la producción de software en las SSE de Chile.

1.2. Conceptos involucrados

Con el fin de establecer cómo se realizará el análisis de costo efectividad de la adaptación automática de procesos, se explicarán los conceptos que permitirán entender con mayor claridad la tesis. Se revisarán los conceptos proceso, contexto y adaptación automática de forma sucinta, para luego explicar qué son los procesos predefinidos y cómo se compararán las estrategias.

Un proceso de software es un conjunto de actividades relacionadas que tiene por objetivo la producción de un producto de software [79]. Definir un proceso requiere especificar sus elementos constituyentes, i.e., tareas, roles y productos de trabajo, así como su orden. Los procesos considerados en esta tesis fueron formalizados utilizando el metaproceso CASPER, un proceso para definir y formalizar procesos de software [36], que utiliza SPEM 2.0 [61] como estándar para la especificación.

Un contexto es un conjunto de atributos y valores que representan el escenario de un proyecto, e.g., Tipo de proyecto con el valor Desarrollo y Tamaño del proyecto con el valor Grande. Una empresa define un contexto organizacional, con el objetivo de instanciar contextos específicos al establecer los atributos y valores relevantes para sus proyectos. Asimismo, un contexto organizacional representa las áreas lógicas más importantes de la empresa, e.g., Proyecto, Equipo, Sistema. Lo anterior se basa en la idea de que cada contexto debe guiar la definición del proceso que mejor se ajusta al proyecto [39].

La adaptación automática considerada en esta tesis [39] se basa en técnicas de *Model-driven engineering* (MDE) [75]. Tanto el proceso organizacional como el contexto son modelos definidos formalmente, entonces el proceso adaptado es posible generarlo mediante una transformación de modelos. Por como ha sido definido y construído el proceso organizacional, las adaptaciones sucesivas con cualquier contexto son obtenidas rápidamente y con poco esfuerzo, más aún óptimas para el contexto. Esta consideración se basa en la exhaustiva investigación realizada en el proyecto ADAPTE [9, 10, 36, 37, 38, 39, 40, 63, 73, 76]. La Ilustración 1.1 muestra un esquema de la adaptación automática.

La adaptación es una transformación de modelos compuesta por reglas de transformación implementadas usando ATL (Atlas Transformation Language) [42], que tiene como entradas un proceso organizacional y el contexto de un proyecto. Un proceso organizacional además de componerse de elementos de proceso posee reglas de variabilidad, que son expresiones lógicas formadas a partir de los atributos y valores del contexto y asociadas a sus elementos. Adaptar un proceso organizacional al contexto de un proyecto significa resolver dichas variabilidades, es decir, determinar los elementos variables a mantener o eliminar.

Por otra parte, los procesos predefinidos están inspirados en la metodología Crystal [23], la que define un conjunto de procesos que son elegidos con respecto a dos atributos: *criticality* y *team size*. Primero para los tipos de proyectos más comunes de la empresa se definirán

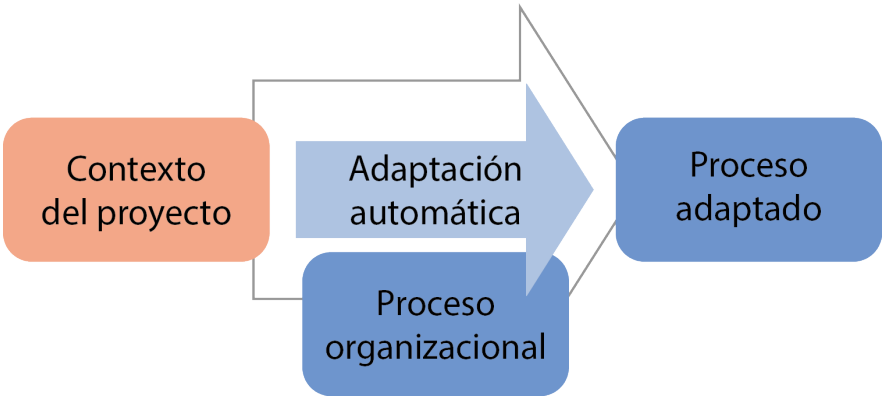


Ilustración 1.1: Adaptación automática.

los contextos correspondientes y se obtendrán sus procesos, generando los pares contexto-proceso. Posteriormente, para adaptar un contexto inesperado se compara dicho contexto con los contextos asociados a los procesos predefinidos; si existe un contexto donde todos los valores coinciden, entonces dicho proceso predefinido es óptimo y es aplicado en el proyecto. En otro caso, se aplica el proceso con el contexto de mayor similitud. La Ilustración 1.2 muestra un esquema de la adaptación de los procesos predefinidos.

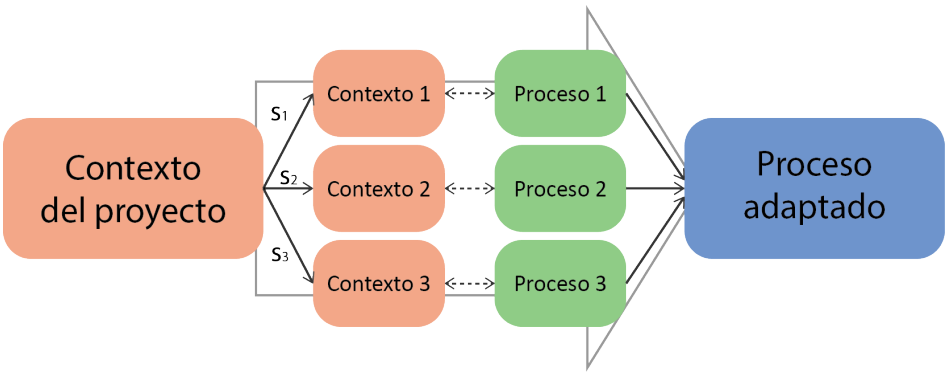


Ilustración 1.2: Adaptación de procesos predefinidos.

Similitud es un valor numérico, resultado de comparar dos contextos y representa lo parecidos que son sus valores: 0 si ambos contextos poseen valores distintos para todos sus atributos, 1 si son todos iguales y un valor intermedio sólo si algunos valores son iguales. Al calcular las similitudes entre el contexto de un nuevo proyecto y para los procesos predefinidos es posible que estas resulten con el mismo valor de similitud, i.e., esas plantillas serían igualmente adecuadas y es imposible decidir *a priori* cuál entregar como resultado.

Para resolver el escenario anterior se ha elegido el *merge* entre dos procesos como el mecanismo que permita refinar la selección del proceso. El *merge* se realizará sólo cuando existan dos similitudes iguales, o bien cuando sean parecidas bajo un umbral definido. Este mecanismo es utilizado para refinar el proceso resultante de la selección, bajo la conjetura de que si para un proyecto existen dos procesos predefinidos diferentes y aproximados, entonces el *merge* entre ellos sería más ajustado a la solución óptima. Esto se deberá verificar bajo el supuesto de que los procesos automáticamente adaptados son óptimos.

Las estrategias de adaptación automática y procesos predefinidos son Software Process Lines. Sus niveles de sofisticación varían junto a los resultados obtenidos, pero los procesos obtenidos en cada caso comparten tareas, roles y productos de trabajo que satisfacen las necesidades específicas de la empresa y son desarrollados desde un conjunto común de elementos de una forma preestablecida [2].

Como se ha mencionado, las estrategias presentadas realizan sus adaptaciones al resolver variabilidades de forma automática u obteniendo la mayor similitud y seleccionando el proceso más adecuado; pero poseen un rasgo en común: toman un contexto y luego efectúan la adaptación. Por lo tanto, comparar ambas estrategias contempla definir un conjunto de contextos inesperados, adaptar dichos contextos en cada caso y luego comparar correspondientemente los procesos adaptados. Esta última comparación evalúa el número de tareas extras y tareas faltantes del proceso predefinido con respecto al proceso automáticamente adaptado.

Con el fin de validar la solución presentada se han considerado independientemente a dos SSE chilenas: Rhiscom y Mobius. Ambas empresas poseen sus procesos organizacionales formalizados y son capaces de obtener procesos utilizando la estrategia de adaptación automática. Por una parte el proceso organizacional de Rhiscom posee 36 tareas, donde 13 de ellas (35, 11 %) son variables, i.e., pueden aparecer o no en el proceso adaptado según los valores del contexto. Para Mobius la cantidad de tareas en su proceso organizacional alcanza el valor de 106 y sus tareas variables corresponden a 42 de ellas (39,62 %). Dado los contextos organizacionales, para Rhiscom la cantidad de contextos posibles es de 135, mientras que para Mobius alcanza la cifra de 2304 contextos.

Dada la gran variabilidad de los procesos organizacionales y la cantidad de contextos posibles para cada empresa, surge la necesidad de caracterizar los procesos predefinidos según los tipos de proyectos más comunes, i.e., un conjunto de los pares contexto-proceso, utilizando *observer triangulation* [74]. Un investigador participó activamente en la definición de los procesos organizacionales, mientras que otro profundizó en los tipos de proyectos más comunes de la empresa para establecer las variabilidades. Se han definido cinco contextos inesperados para cada empresa que representan casos posibles que éstas puedan enfrentar.

1.3. Hipótesis

Bajo la premisa de que los procesos obtenidos mediante la adaptación automática son óptimos para cualquier contexto, las hipótesis de esta tesis son:

Hipótesis 1: *Los procesos predefinidos serán igual de adecuados a la adaptación automática, cuando los contextos inesperados correspondan exactamente a los procesos predefinidos.*

Hipótesis 2: *Una variación en el contexto impacta tanto en productividad y calidad al proceso resultante.*

Hipótesis 3: *Al realizar un* merge *entre los procesos predefinidos de mayor similitud, bajo un umbral definido, se obtendrá un proceso más similar al óptimo.*

1.4. Objetivos

El objetivo general de esta tesis es comparar los resultados de dos estrategias de adaptación de procesos, para obtener datos cuantitativos de la mejora presentada al adoptar la estrategia de adaptación automática de procesos con respecto la estrategia utilizada anteriormente por las SSE, un conjunto de procesos predefinidos. Es posible enumerar los objetivos específicos como sigue:

1. Definir la arquitectura correspondiente para una estrategia basada en plantillas.
2. Construir un seleccionador de procesos en función de la mínima similitud entre contextos.
3. Construir un comparador de procesos que establezca la cantidad de tareas extras y tareas faltantes.
4. Implementar el *merge* entre procesos predefinidos para evaluar el punto anterior.
5. Determinar cuándo es apropiado utilizar los procesos predefinidos para minimizar la cantidad de tareas extras y/o las tareas faltantes.
6. Identificar las amenazas de validez presentes en la realización de esta tesis.

1.5. Metodología

El problema derivado de la oportunidad planteada no ha experimentado cambios durante la realización del trabajo; sin embargo la solución tuvo que pasar por diferentes fases antes de ser construida y evaluada. El trabajo fue realizado en cuatro etapas:

1. **Adquisición de conocimiento:** El objetivo de esta etapa fue entender el escenario y las restricciones del problema. Se revisó la investigación realizada por ADAPTE, en particular la relacionada a la adaptación automática junto a otras propuestas alternativas. Se inspeccionaron otras estrategias de solución y sus fundamentos teóricos, para finalmente decidir por los procesos predefinidos.
2. **Piloto conceptual:** Paralelo a lo anterior, se realizó un piloto conceptual que permitió validar si existen y son medibles los conceptos de similitud, tareas extras y tareas faltantes. Asimismo se evaluaron los posibles riesgos de efectuar la adaptación con los procesos predefinidos.
3. **Desarrollo de prototipo:** También en paralelo, este paso contempló dos procedimientos: una selección de procesos predefinidos y la comparación entre estrategias. En ambas instancias se evaluó utilizar herramientas que permitieran realizar estos procedimientos de forma automática, e.g., EMF Compare, SiDiff, pero resultaron no ser adecuadas para este caso particular. Se decidió desarrollar una solución a medida.

4. **Experimentación y mejora:** Luego de realizar el desarrollo del prototipo y validarlo, se utilizaron los procesos de Rhiscom y Mobius para evaluar los resultados de las adaptaciones automáticas y los procesos predefinidos. Al obtener los primeros resultados sobre la calidad y productividad se consideró realizar un *merge* y evaluar su aporte, bajo la conjetura de que los resultados se asemejarían más a un proceso automáticamente adaptado.

1.6. Estructura de esta tesis

La presentación del informe de este trabajo de tesis, "Elección entre procesos automáticamente adaptados y procesos predefinidos", se realiza mediante seis capítulos cuyo contenido se expone a continuación:

1. En el primer capítulo, se ha evaluado la situación actual de las SSE en Chile y cómo ADAPTE ha presentado una estrategia de adaptación de procesos que es factible y usable. Fue presentada la oportunidad, la hipótesis que se busca probar, la solución propuesta, sus objetivos generales y específicos junto a la metodología de trabajo a utilizar.
2. El segundo capítulo presenta el marco teórico en que se apoya este trabajo, los conceptos utilizados y su explicación en profundidad, los modelos utilizados para procesos y contextos, las soluciones disponibles para realizar adaptación de procesos y algunas de las alternativas para realizar la comparación entre modelos.
3. El tercer capítulo presenta el problema u oportunidad que presenta esta tesis; además se presenta la relevancia de una solución.
4. El cuarto capítulo muestra el diseño de la solución junto a su implementación. En primera instancia se explica la arquitectura utilizada para ambas estrategias y posteriormente la implementación de dos módulos principales: selección de procesos predefinidos y comparación entre estrategias.
5. El quinto capítulo presenta taxonomías para el proceso y el contexto organizacional, además de mostrar la validación experimental realizada con la información de cada SSE. Posteriormente se analizan los resultados obtenidos.
6. El sexto capítulo enuncia las conclusiones y contribuciones de esta tesis, junto a las amenazas a la validez y propuestas de trabajo futuro.

Capítulo 2

Marco Teórico

En este capítulo se presentan las bases teóricas relevantes del problema. En primer lugar, se revisarán los conceptos utilizados explicándolos dentro del alcance de la tesis. Se presentan las diferentes soluciones encontradas en la literatura y las justificaciones de por qué son o no son adecuadas para resolver el problema planteado. Lo anterior permitirá justificar las decisiones de diseño y la definición concreta de la solución.

2.1. Modelos y metamodelos

Model-Driven Engineering (MDE) es una metodología de desarrollo de software que se basa en la premisa de que "*todo es un modelo*" [12]. Un modelo es una representación abstracta de aquellos aspectos del objeto o fenómeno que son, o se considera que son, relevantes para su entendimiento y propósito [57], más aún, constituyen la base y la única fuente de información para especificar y generar un sistema [16]. Un lenguaje de modelado se utiliza para definir formalmente un modelo, por lo tanto, permite al artefacto ser operable mediante transformaciones según la sintaxis y semántica del lenguaje. En particular, uno de los objetos a tratar en esta tesis es un modelo de procesos de software.

Un metamodelo define el lenguaje de modelado, compuesto por los elementos y reglas necesarias para construir un modelo. Se dice que un modelo conforma con un metamodelo si el primero fue instanciado a partir del último. Asimismo, se define como metametamodelo al lenguaje de modelado utilizado para definir un metamodelo; más aún, un metametamodelo conforma con sí mismo. En la Ilustración 2.1 se muestra la Organización 3+1 propuesta por Bézivin [13]: un modelo (M1) es una representación simplificada de lo que llamamos sistema (M0) en el mundo real, mientras que dicho modelo conforma con un metamodelo (M2) y este a su vez con un metametamodelo (M3).

El proceso de desarrollo en MDE está compuesto de una serie de transformaciones sobre modelos. Usualmente, las transformaciones son refinamientos sobre los modelos que dismi-

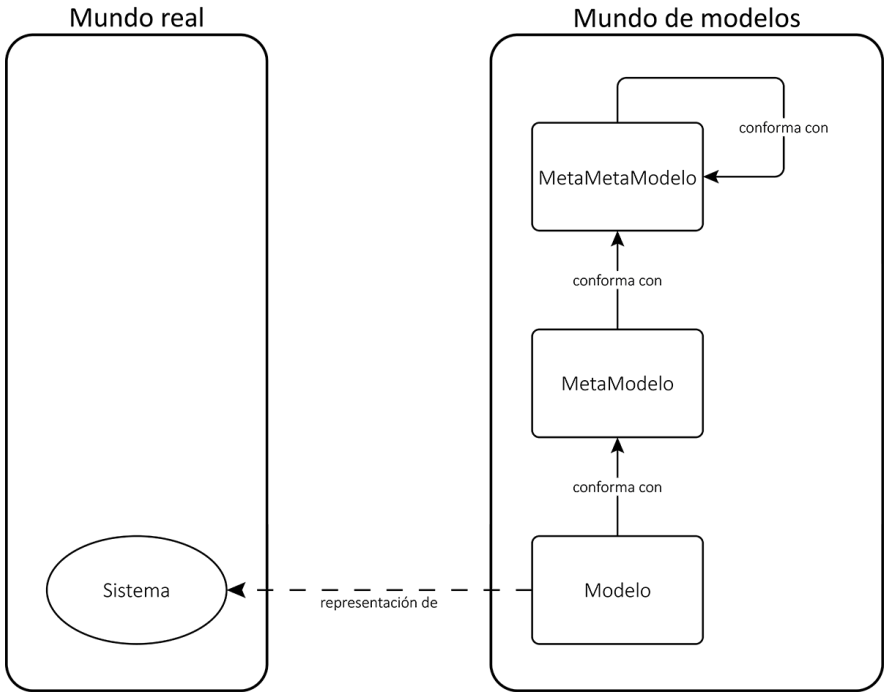


Ilustración 2.1: Organización 3+1 [13].

nuyen el nivel de abstracción, con el objetivo de producir un modelo que posea suficientes detalles para la generación automática de código ejecutable, para realizar *refactoring* o ingeniería inversa [42]. Una transformación considera un modelo fuente, un modelo objetivo y una definición de transformación. Kleppe et al. [46] otorgan una definición que incluye estos conceptos: "Una transformación de modelos es la generación automática de un modelo objetivo desde un modelo fuente, de acuerdo con una definición de una transformación. Una definición de una transformación es un conjunto de reglas de transformación que juntas describen cómo un modelo en el lenguaje de la fuente puede ser transformado en un modelo en el lenguaje objetivo. Una regla de transformación es una descripción de cómo uno o más constructos en el lenguaje fuente pueden ser transformados en uno o más constructos en el lenguaje objetivo." Un ejemplo específico es la transformación utilizada por la adaptación automática de procesos propuesta por ADAPTE, para configurar un proceso a las particularidades de un proyecto.

Mens et al. [56] distinguen las transformaciones según los modelos de fuente y objetivo, como también el nivel de abstracción de estos modelos. Una transformación donde el lenguaje de los modelos es el mismo tanto para el modelo fuente como para el modelo objetivo, es descrita como una transformación endógena; mientras que si los lenguajes de los modelos son distintos, es una transformación exógena. Ejemplos de transformaciones endógenas son *refactoring* de software y la adaptación de componentes, al modificar el código de un software existente; para una transformación exógena, es la generación de bytecode desde código fuente. Si los modelos de fuente y objetivo residen en el mismo nivel de abstracción, entonces se dice que la transformación es horizontal; mientras que si los modelos poseen diferentes niveles de abstracción, es una transformación vertical. Un ejemplo de transformación horizontal es la migración de un software desde un lenguaje a otro; y para la transformación vertical es el

refinamiento, en pasos sucesivos, para agregar detalles concretos. En particular, la adaptación automática de procesos utiliza una transformación que es endógena y vertical. La Tabla 2.1 muestra las dimensiones ortogonales de las transformaciones de modelos y sus ejemplos.

	Horizontal	Vertical
Endógena	*Refactoring*	Refinamiento
Exógena	Migración	Generación de código

Tabla 2.1: Dimensiones ortogonales de las transformaciones de modelos [56].

En la industria, las estrategias basadas en MDE son consideradas atractivas por su mayor calidad, eficiencia, facilidad de predicción y escalabilidad. Pueden ser utilizadas de diferentes maneras: para predecir la calidad de un sistema, para evaluar propiedades específicas cuando el sistema sufre cambios o para comunicar características claves a los *stakeholders* [11]. A continuación se expondrán las herramientas y modelos que forman parte de los objetos principales de esta tesis: procesos, contextos y su relación en una adaptación.

2.2. Procesos

Un proceso de software es, de acuerdo con Osterweil *"cualquier mecanismo utilizado para llevar a cabo o lograr un objetivo de software de forma ordenada"* [64], mientras que para Sommerville es *"un conjunto de actividades relacionadas que conduce a la producción de un producto de software"* [79]. Estas definiciones transmiten el objetivo principal de un proceso de software, que es guiar las actividades de desarrollo de software en una empresa. En una SSE, un proceso de software puede representar diferentes tipos de actividades como la extensión o modificación una herramienta existente, una configuración e integración con otros componentes o la realización de una solución a medida.

Un proceso de software es la realización de un modelo de procesos de software, o simplemente modelo de procesos. Dicho modelo encapsula el *know-how* de la empresa al tener la posibilidad de representar todos sus procesos. Ahora bien, Münch ofrece una caracterización clara de lo que es un modelo de procesos, pero que en español resulta ser idempotente:

"A software process model is a model of a software process" [57].

Anteriormente, se ha mencionado que la adaptación automática obtiene procesos de software para guiar el desarrollo de proyectos concretos mediante el refinamiento de un modelo de procesos; para esta tesis se considera que este modelo conforma con eSPEM, que es el lenguaje de modelado o metamodelo de procesos desarrollado por ADAPTE para este fin. eSPEM es un subconjunto de SPEM 2.0, i.e., conforma con este último, y más aún, considera sólo los mecanismos de variabilidad que resultan ser suficientes en la práctica [39]. En el contexto de ADAPTE, la formalización del modelo de procesos sigue los pasos indicados por Ruiz et al. [73], utilizando la plataforma EPF Composer[1] como apoyo en la definición y publicación de los procesos. A continuación se profundizan los elementos mencionados.

[1]http://eclipse.org/epf/ - Última visita: 18 de Diciembre 2014

2.2.1. SPEM 2.0

Actualmente los equipos que se desempeñan en el desarrollo de software poseen la capacidad de comunicarse y acceder a la información por diferentes canales, lo que presenta la necesidad de un lenguaje común que otorgue permanencia y uniformidad a la información transmitida, y que actúe en la misma dirección que las prácticas utilizadas. Con este objetivo el Object Management Group[2] (OMG) ha definido Software & Systems Process Engineering Meta-Model 2.0[3] (SPEM 2.0) con su última versión publicada en abril del 2008. Toda implementación de SPEM 2.0 tiene por objetivo representar, utilizar y administrar librerías que posean contenido reutilizable para el desarrollo de procesos de software. Su uso está dirigido, pero no restringido a: ingenieros de procesos, jefes y administradores de proyectos que sean responsables de mantener e implementar procesos de desarrollo en una empresa.

SPEM 2.0 es un metamodelo que conforma con Meta Object Facility (MOF) [62], i.e., su metametamodelo, el cuál provee una base para la definición del lenguaje de modelado. Si bien SPEM 2.0 describe todas las estructuras y atributos necesarios para representar procesos, no posee una manera nativa de expresar gráficamente su propia información. Para esto extiende el metamodelo de UML 2 como *profile*, i.e., como una extensión personalizada de los modelos de UML para este dominio particular. Una *Method Library* provee instancias concretas de sus elementos para estructurar un proceso de software. La Ilustración 2.2 muestra la organización con respecto a los niveles de modelado para SPEM 2.0 y UML [61], estructurado según la arquitectura de MDE propuesta por Bézivin [13].

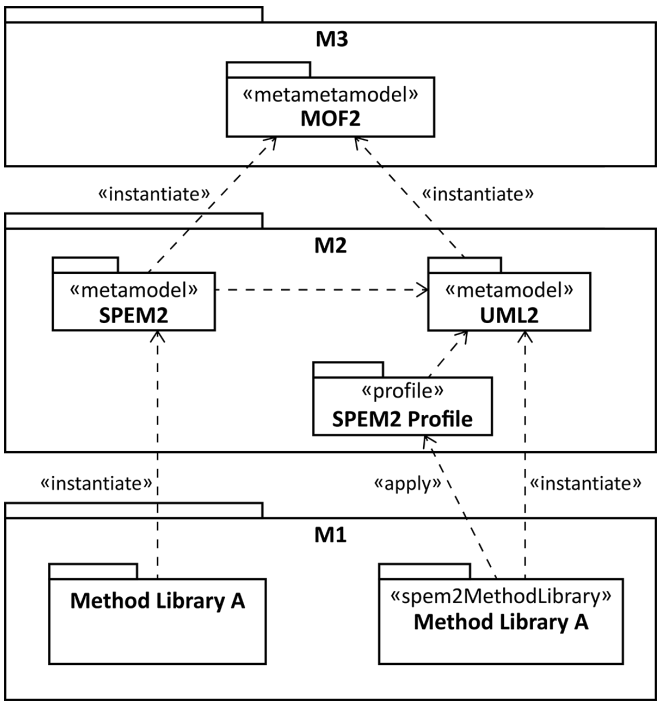


Ilustración 2.2: Niveles de modelado para SPEM 2.0 y UML [61].

[2]http://www.omg.org/

[3]http://www.omg.org/spec/SPEM/2.0/ - Última visita 18 de Diciembre 2014

Asimismo, SPEM 2.0 define una estructura llamada *Method Library* que puede contener las descripciones de tareas, roles y productos de trabajo, junto con sus relaciones. La Ilustración 2.3 presenta el modelo conceptual de SPEM 2.0: una o más tareas son ejecutadas por uno o más roles, mientras que consume o genera uno o más productos de trabajo, y cada rol es responsable de uno o más productos de trabajo.

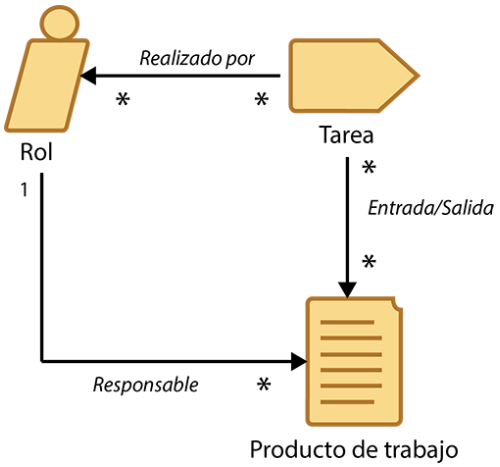


Ilustración 2.3: Modelo conceptual de SPEM 2.0.

La Ilustración 2.4 presenta la estructura del metamodelo SPEM 2.0. Está estructurado en siete *metamodel packages*, donde la dependencia esta representada según el mecanismo de *merge* de UML 2. Una instancia de este metamodelo es generada al agregar o extender los *packages* gradualmente para proveer las capacidades deseadas al modelo.

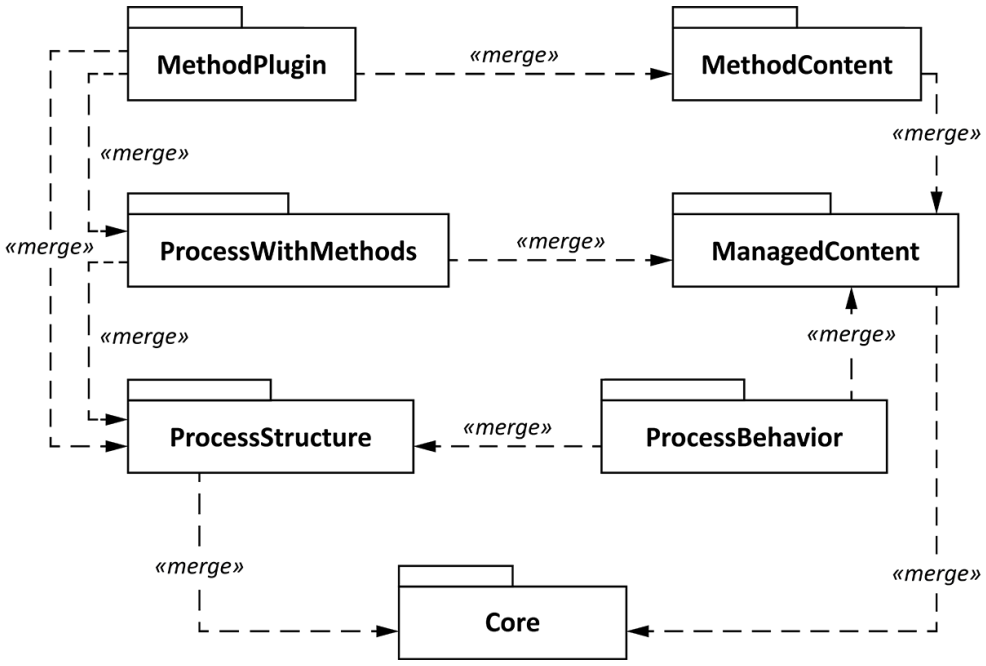


Ilustración 2.4: Metamodelo de SPEM 2.0 [61].

Los *metamodel packages* mostrados en la Ilustración 2.4 son [61]:

1. ***Core***: Este *package* contiene todas las clases y abstracciones del metamodelo que constituyen la base para todas las otras clases de los *metamodel packages* restantes.
2. ***Process Structure***: Este *package* define la base para todos los modelos de proceso. Su estructura está basada en actividades y las respectivas clases de roles que las ejecutan, además de las entradas y salidas como clases de productos de trabajo para cada actividad. Las estructuras descritas permiten representar tanto procesos de alto nivel como básicos, pero sin documentación textual.
3. ***Process Behavior***: Este *package* permite extender, mediante modelos de comportamiento, los elementos del *package Process Structure*. Sin embargo, no define su propio modelo de comportamiento, sino que provee "enlaces" a modelos definidos externamente para permitir la reutilización por parte de otras especificaciones de la OMG. Por ejemplo, un proceso definido según el *package Process Structure* puede ser enlazado a diagramas de actividad de UML 2 para representar su comportamiento.
4. ***Managed Content***: Este *package* introduce los conceptos para administrar la documentación en lenguaje natural de las descripciones del proceso. Estos conceptos pueden ser utilizados por sí solos o en combinación con los conceptos del *package Process Structure*.
5. ***Method Content***: Este *package* provee los conceptos para los usuarios y organizaciones de SPEM 2.0, que deseen construir una base de conocimiento base para el desarrollo que sea independiente del desarrollo de procesos y proyectos particulares. Para lo anterior, permite definir elementos reusables que provean una base para documentar el conocimiento sobre métodos de desarrollo, técnicas y realizaciones concretas de las prácticas utilizadas. Estos elementos están compuestos de explicaciones paso-a-paso que describen cómo las metas específicas son logradas; por cuáles roles, con qué recursos y resultados; además de ser independientes dentro de un ciclo de desarrollo particular.
6. ***Process With Methods***: Este *package* define nuevas estructuras y redefine otras existentes para integrar procesos definidos según el *package Process Structure*, para integrarlas con los conceptos correspondientes a *Method Content*. *Method Content* al definir los métodos y técnicas fundamentales para el desarrollo de software, permite representar los procesos que otorgan lugar a dichos elementos dentro del contexto de ciclo de vida del modelo mediante fases o hitos.
7. ***Method Plugin***: Este *package* introduce el concepto de "diseño" y la administración de librerías o repositorios de *Method Content* o procesos, que deben ser mantenibles, escalables, reusables y configurables. Con conceptos tales como *Method Plug-in*, *Process Component* y *Variability*, es posible definir procesos con mayores capacidades.

Por ejemplo, es posible caracterizar subconjuntos de los *metamodel packages* y sus usos recomendados:

- *Process Structure* y *Process Behavior*: Sólo modelado, sin documentación.
- *Process Structure* con *Managed Content*: Para documentar como también para publicar los procesos.
- *Process Structure* con *Method Plugin*: Para administrar conjuntos de procesos que extienden unos de otros.
- Todo excepto *Method Plugin*: Procesos completamente documentados con su contenido, pero a pequeña escala.

Finalmente, el uso de una implementación de SPEM 2.0 permite dentro de un *framework* conceptual [61]:

- **Proveer librerías de *Method Content* reusables junto a una representación estandarizada**: Los desarrolladores necesitan entender los métodos y técnicas claves utilizadas en la ejecución de su trabajo, e.g., obtener y administrar requerimientos, efectuar el análisis y diseño, implementar diseños o casos de prueba, probar una implementación contra los requerimientos, administrar el alcance del proyecto y el cambio, entre otras. Asimismo, necesitan entender cuáles tareas producen ciertos productos de trabajo, además de las habilidades que son requeridas. SPEM 2.0 apunta a ayudar a los practicantes de software en la creación de un capital intelectual de conocimiento base para el desarrollo de software y sistemas que les permiten administrar y desplegar su contenido utilizando un formato estándar. Tal contenido puede consistir en cualquier descripción general de cómo desarrollar software dentro de una empresa, e.g., guías, plantillas, principios, prácticas, procedimientos o regulaciones internas. Este conocimiento base puede ser usado como referencia para nuevas personas.
- **Apoyar el desarrollo, administración y crecimiento sistemático de procesos de desarrollo**: Gracias al ciclo de vida de un proyecto, los equipos definen cómo aplicar sus métodos y prácticas de desarrollo, i.e., necesitan definir o seleccionar un proceso de desarrollo. Más aún, es necesario que el proceso pueda expresar diferencias tales como administrar requerimientos durante distintas fases del proyecto o si el proyecto es nuevo o es la mantención de un sistema existente. Es por lo anterior que SPEM 2.0 permite crear procesos basados en *Method Content* reusable, mientras que provee las bases para que ingenieros de procesos y jefes de proyectos puedan seleccionar y adaptar procesos.
- **Apoyar el uso de los contenidos necesarios para definir la configuración del proceso y *Method Content***: En SPEM 2.0, las nociones de *Activity Use*, *Configurability* y *Variability* para procesos de desarrollo y *Method Content*, están dirigidas exactamente a las necesidades de los procesos definidos. Un proceso definido es un proceso administrado, i.e., realiza actividades que pueden ser reconocidas como implementaciones de prácticas de desarrollo, que además es adaptado desde un conjunto de procesos estándares de la empresa de acuerdo con sus propias guías de adaptación. De esta forma, las nociones mencionadas permiten a un proceso definido reutilizar otros procesos o patrones de procesos para modelar diferentes elementos variables y para adaptar procesos generales.
- **Apoyar la promulgación de un proceso para el desarrollo de proyectos**: La definición de un proceso sólo entrega valor si impacta y dirige el comportamiento de los equipos de desarrollo, ya sean jefes de proyectos o desarrolladores, es necesario que el proceso y el *Method Content* esté disponible. Además, es necesario que el proceso esté disponible en un formato adecuado a la plataforma utilizada por el equipo para la promulgación, e.g., sistemas de *backlog*, motores de *workflow*, sistemas de administración de recursos. SPEM 2.0 provee estructuras de definición de procesos que permiten a los ingenieros de procesos expresar la manera en que el proceso será promulgado dentro de estos sistemas.

2.2.2. Software Product Lines y Software Process Lines

El concepto de Software Product Lines (SPL) nace desde la ventaja de que los diferentes elementos de software producido en una organización poseen aspectos comunes, e.g., cálculos, plataformas o características. Así, Clements y Northrop definen SPL como:

"A Software Product Line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [21].

Al establecer que un producto es desarrollado por un equipo, la reutilización sistemática del código y librerías es pequeña. Sin embargo, al utilizar un conjunto común de elementos que sólo debe ser desarrollado una vez, es posible reducir el esfuerzo involucrado en su desarrollo y su mantención. SPL es posible describirlo como el siguiente conjunto de pasos [2]:

- **Product Line Scoping**: En este paso se define el alcance de la SPL al identificar el conjunto de características reusables que los sistemas dentro de la SPL deben poseer.
- **Product Line Modeling**: En este paso se analizan las características dentro de la SPL para buscar elementos comunes y elementos variables.
- **Product Line Architecting**: En este paso se establecen referencias que corresponden a los resultados de los pasos anteriores, e.g., una tabla de decisión que correlaciona las variabilidades con elementos de la SPL.
- **Product Line Instantiation**: Una vez completa la construcción, es posible instanciar productos específicos desde la SPL, i.e., se debe resolver la tabla de decisión generada y posteriormente agregar las características faltantes.

De forma análoga, los pasos antes descritos para SPL pueden ser transferidos a Procesos de software [2]:

- **Process Line Scoping**: En este paso, los productos y proyectos—actuales y futuros—de la empresa son analizados para establecer las necesidades de los procesos. Estas necesidades forman el *scope* de la Process Line.
- **Process Line Modeling**: En este paso se analizan los procesos en términos de elementos comunes y variables, lo que obtiene una colección de elementos de procesos genéricos de los cuáles es posible construir un modelo de decisión.
- **Process Line Architecting**: En este paso se crea un modelo de procesos que contiene un conjunto común y los elementos variables, el cuál es instanciado en procesos individuales usando el modelo de decisión generado en el segundo paso.
- **Process Line Instantiation**: En este paso es posible obtener un proceso para un proyecto con características específicas.

De forma correspondiente a [21], Armbrust et al. definen una *Software Process Line* (SPrL) como:

"a set of software processes with a managed set of characteristics that satisfy the specific needs of a particular organization and that are developed from a common set of core processes in a prescribed way" [2].

De forma análoga, Washizaki define el concepto de *Process Line* que define como:

"a set of processes in a particular domain or for a particular purpose, having common characteristics and built based upon common, reusable process assets" [86].

Más aún, el concepto de *Process Line Architecture*:

"a process structure which reflects the commonality and variability in a collection of processes that make up a process line from the perspective of overall optimization" [86].

Washizaki relaciona ambos conceptos al establecer que una *Process Line Architecture* (PLA) es generada al abstraer una *Process Line*, lo que resume como una *"overall optimization"*; que es preparar una PLA con suficiente generalidad para generar obtener procesos individuales que permitan generar una *Process Line* mediante un esfuerzo acotado.

Los procesos obtenidos desde una PLA son específicos para un proyecto, pero sus elementos constituyentes pueden ser combinados, extendidos o reutilizados para un proyecto similar. Por otro lado, se utiliza un enfoque *bottom-up* para generar una PLA al clasificar los elementos que forman un proceso principal, lo que requiere obtener los elementos desde la *Process Line*. Para generar u obtener procesos desde una *Process Line* o *Software Process Line*, tanto por Armbrust et al. como por Washizaki han definido variabilidades asociadas a los elementos del proceso. Se tienen tres tipos de variabilidades [86]:

- **Comunes o Mandatorias**: Una tarea común siempre está presente en los procesos de la *Process Line* y, por lo tanto, no posee variabilidad asociada.
- **Puntos de variabilidad u Opcionales**: Los puntos de variabilidad son elementos, i.e., tareas roles o productos de trabajo, que pueden ser cambiados de acuerdo con las características de un proyecto en específico.
- **Procesos variables o Alternativas**: Un proceso variable puede ser reemplazado por otros elementos según las características del proyecto en específico.

El mecanismo de variabilidad asociado a los elementos de procesos está reflejado en el metamodelo de procesos que utiliza la adaptación automática: eSPEM.

2.2.3. Metamodelo de procesos: eSPEM

Anteriormente se presentaron las características relevantes de SPEM 2.0. Entre ellas se encuentra la capacidad de definir un subconjunto de *metamodel packages*, como una implementación que se adecúe a las necesidades del problema. Gracias a lo anterior, dentro del marco del proyecto ADAPTE se ha generado un modelo que es un subconjunto de SPEM 2.0 llamado *experimental* SPEM (eSPEM) [35], el cuál es suficiente para los propósitos de dicha investigación. La Ilustración 2.5 presenta el metamodelo de procesos eSPEM.

eSPEM ha sido modelado como un *Method Plug-in* que incluye los Process Elements y sus *Method Content Elements* asociados. Los *Method Content Elements* corresponden a *Task*

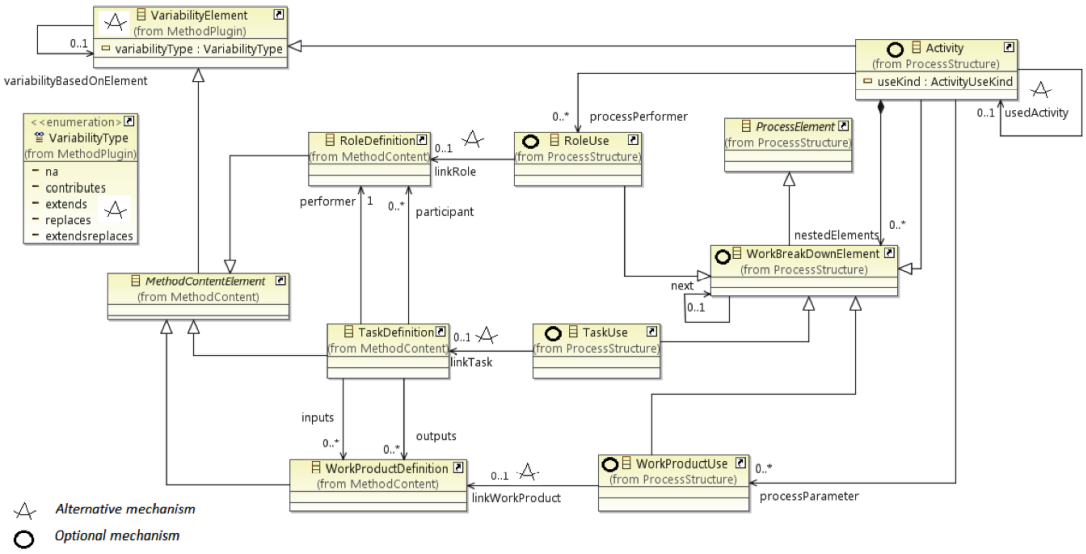


Ilustración 2.5: Experimental SPEM - (eSPEM) [35]

Definitions que son realizadas por *Role Definitions* y poseen *Work Product Definitions* como entradas y salidas. Una *Activity* es un *Work Breakdown Element* y un *Work Definition*, que define unidades de trabajo dentro de un proceso como también un proceso mismo. Además, una *Activity* permite anidar y agrupar lógicamente otros *Work Breakdown Elements*, permitiendo su reutilización como una subestructura. De esta forma, *Task Use*, *Role Use* y *Work Product Use* son *Work Breakdown Elements* que se refieren a las ocurrencias específicas dentro de una *Activity* de su respectivo *Method Content Element* [35].

En eSPEM, un *Variability Element* es un elemento que puede ser modificado o extendido por otro *Variability Element* del mismo tipo de acuerdo con un *Variability Type* (e.g., *extends, replaces*, *contributes*, *extends-replace*) [61]. De esta forma, cada *Method Content Element*, i.e., *Task Definition*, *Role Definition* y *Work Product Definition*, y las metaclases de *Activity* son *Variability Elements*. Es posible utilizar *Variability Elements* cuando variantes alternativas lo requieran y una condición general o una restricción deban ser cumplidas. Es posible definir un conjunto de alternativas desde un *Variability Element* en particular y seleccionarlas si posee un *Process Element* asociado. Más aún, cada *Work Breakdown Element* puede ser considerado como opcional o no [35].

Ya que todo *Work Breakdown Element* puede tener asociado un *Variability Element*, este último permite articular el procedimiento de adaptación automática, más aún, las instancias de eSPEM tienen la capacidad de ser *adaptables* al permitir su configuración. Finalmente, eSPEM al ser un subconjunto de SPEM 2.0, está compuesto por los siguientes *metamodel packages* [35]: *Process Structure*, *Method Content*, *Process with Methods* y *Method Plugin*.

2.2.4. Metaproceso CASPER

Uno de los problemas principales que enfrentan los ingenieros de software es desarrollar tanto software como procesos de software, enfrentando un tiempo ajustado. Para que la adaptación de un proceso de software consuma una menor cantidad de tiempo, para el ingeniero y los *stakeholders* además de promover la reutilización del conocimiento entre los proyectos de la misma organización, ADAPTE genera el metaproceso *Context Adaptable Software Process EngineeRing* (CASPER) [36], que tiene por objetivo generar la infraestructura necesaria para efectuar una adaptación automática en una SSE.

Para una efectiva adopción de CASPER, la organización debe contar con dos equipos: el *Software Process Engineering Group* (SPEG) y el *Project Team* (PT). Estos equipos ejecutan los respectivos subprocesos [36]:

- ***Process domain engineering***: Como se presenta en la Ilustración 2.6, corresponde a un proceso iterativo que se encarga de capturar el conocimiento sobre el dominio de procesos de software al efectuar cinco actividades: *process context analysis*, *process feature analysis*, *scoping analysis*, *reference process model design* y *production strategy implementation*. Estas actividades son realizadas por ambos equipos, aunque el PT participa al presentar los requerimientos y el conocimiento con respecto a la empresa. El resultado es un modelo de procesos organizacional adaptable a un contexto.
- ***Process application engineering***: Como se presenta en la Ilustración 2.7, define el contexto específico de un proyecto de acuerdo con la información del proyecto además de ejecutar la estrategia de producción. El PT define el contexto más apropiado a la situación del proyecto.

CASPER utiliza eSPEM como *framework* para estructurar los procesos organizacionales. Los pasos básicos para estructurar un proceso que posea variabilidades son [36]:

1. **Crear una *Method Library***: Una *Method Library* es el contenedor del modelo de proceso organizacional de software y de los modelos de procesos de software adaptados.
2. **Identificar *method plug-ins* comunes, opcionales y alternativos**: Es necesario crear y reutilizar los *method plug-ins* respectivos para definir los puntos de variabilidad, tanto en el proceso principal como en sus asociaciones con respecto a los *method plug-ins* variables.
3. **Para cada *Method Plug-in* crear un *Method Content Package***: Se debe estructurar cada *Method Content Package* de acuerdo con las decisiones de diseño. Lo que implica:
   - Crear e identificar los *Method Elements*.
   - Relacionar los *Process Features* a sus respectivos *Method Elements* identificando las variabilidades.
   - Para cada conjunto de variabilidades o puntos de variabilidad, debe ser definida una jerarquía de *Method Elements* utilizando los tipos de variabilidad indicados en eSPEM.

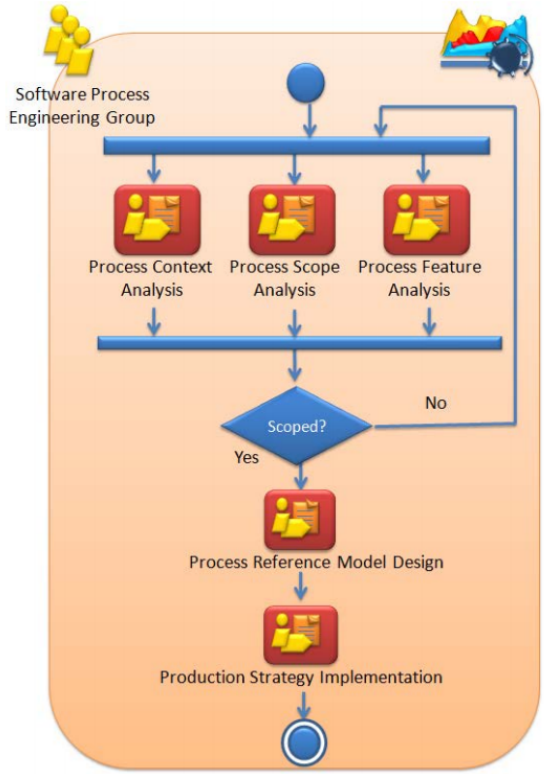


Ilustración 2.6: CASPER *Domain Engineering Meta-Process* [36]

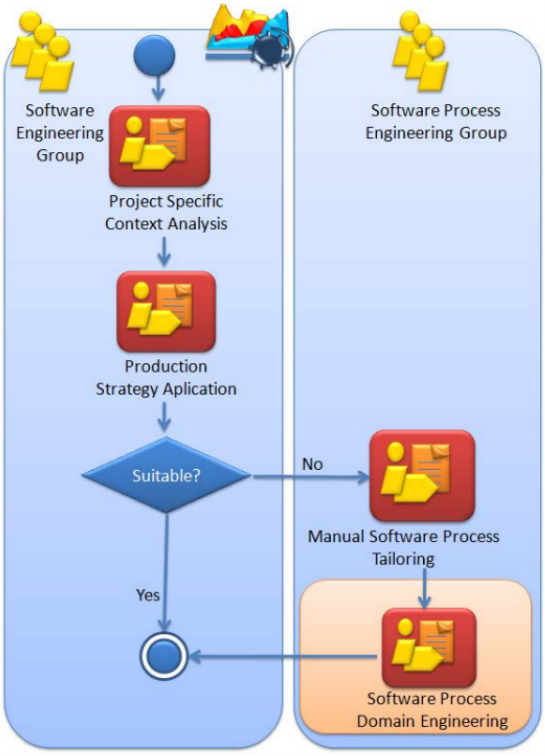


Ilustración 2.7: CASPER *Application Engineering Meta-Process* [36]

4. **Para cada *Method Plug-in*, crear el respectivo *Process Package***: Es necesario estructurar cada *Process Package* de acuerdo con las decisiones de diseño. Esto implica:
   - Crear e identificar los *Process Patterns* principales y variables.
   - Relacionar los *Process Features* a los respectivos *Process Elements*, incluyendo *Process Patterns*.
   - Para cada alternativa identificada que es parte de una *Activity*, una jerarquía de elementos puede ser creada utilizando los tipos de variabilidad indicados en eSPEM.

2.2.5. Eclipse Process Framework

Al desarrollar las actividades en una empresa, son necesarios diferentes medios de comunicación que faciliten el conocimiento sobre las tareas realizadas por los equipos de trabajo. Si bien es clara la necesidad de herramientas específicas que soporten una comunicación formal [49] o informal, estos medios son los componentes principales al desarrollar sistemas de gran escala, y especialmente cuando es necesario cumplir los requerimientos de los usuarios y *stakeholders*.

Acorde a estas necesidades, surge *Eclipse Process Framework*[4] (EPF), un proyecto impul-

[4]http://eclipse.org/epf/ - Última visita: 18 de Diciembre del 2014

sado por la comunidad *open source* Eclipse[5] con dos objetivos principales:

- Proveer un *framework* extensible y herramientas para la ingeniería de procesos de software; creación, administración, configuración y publicación de procesos.
- Proveer modelos de procesos extensibles para el desarrollo y administración de procesos de software de forma iterativa, ágil e incremental, y aplicables a un amplio conjunto de plataformas y aplicaciones.

En particular, el *framework* está compuesto de dos componentes principales:

- Metamodelo: Se utiliza el metamodelo formal *Unified Method Architecture* (UMA), que ha sido especificado en MOF, diagramas UML, como también en *schemas* XML. UMA es una evolución de SPEM 1.1, aunque se espera dar soporte de forma completa a SPEM 2.0 en el futuro cercano.
- Herramientas: La comunidad provee de funcionalidades básicas que pueden ser extendidas de ser necesario. Estas funcionalidades permiten *method authoring*, *process authoring*, *library management*, y *configuring and publishing*.

Con lo anterior, *Eclipse Process Framework Composer*[6] (EPF Composer) es una aplicación de software gratuita, generada desde el proyecto EPF. Asimismo, utiliza el *framework* definido por EPF para promover el aprendizaje de su propia plataforma y el modelamiento de procesos de software. En una empresa, el rol de EPF Composer es:

- Habilitar a ingenieros de procesos, jefes y administradores de proyectos para mantener e implementar procesos de desarrollo.
- Proveer una representación estandarizada y librerías de métodos reusables.
- Administrar el crecimiento sistemático, para definir, publicar y aplicar procesos de desarrollo en las empresas.
- Mantener las prácticas de la empresa a través del ciclo de vida del proyecto.

Ahora bien, EPF Composer no se utiliza para definir la adaptación basada en plantillas propuesta en esta tesis. EPF Composer sí fue utilizado para definir los modelos de procesos en las empresas que participaron ADAPTE.

2.3. Contextos

En esta tesis, las características particulares de un proyecto son un `contexto`. Un contexto es definido según las características del proyecto que representa, e.g., tamaño del equipo, duración del proyecto, complejidad del producto, más aún, puede considerar diferentes aspectos de la empresa. En ADAPTE, dichos aspectos son clasificados según la experiencia de los participantes al formalizar el contexto, que por casualidad está alineado con una propuesta reciente que establece un criterio para efectuar la adaptación de procesos de software según

[5]http://www.eclipse.org - Última visita: 18 de Diciembre del 2014
[6]http://eclipse.org/epf/tool_component/tool_index.php - Última visita: 18 de Diciembre de 2014

el contexto de un proyecto [43]; donde se proponen cuatro grupos: Equipo, Ambiente interno, Ambiente externo y Objetivos.

De igual forma que para los procesos, ADAPTE ha diseñado *Software Process Context Metamodel* (SPCM). SPCM es un metamodelo que permite a las empresas definir su propio modelo de contextos o contexto organizacional, y los contextos para sus procesos de software [35]. La Ilustración 2.8 presenta el modelo y sus relaciones.

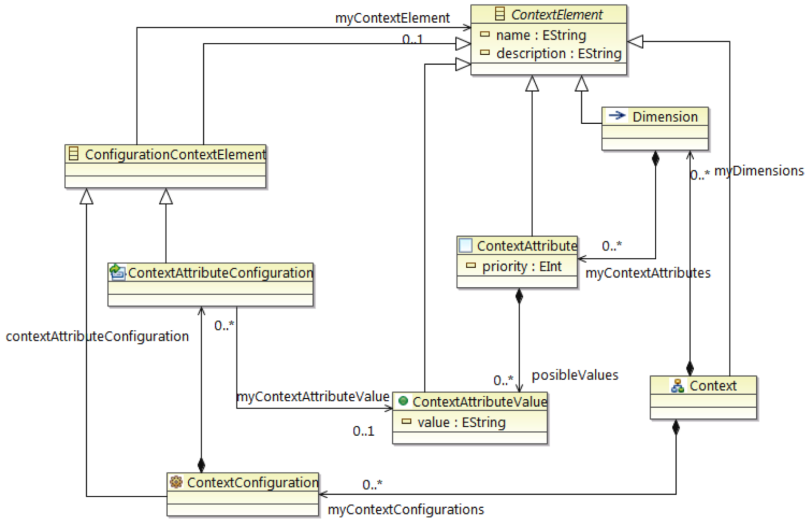


Ilustración 2.8: Software Process Context Metamodel - (SPCM) [39]

En SPCM, todos los elementos descritos a continuación extienden de *Context Element*, que posee un nombre y una descripción. Asimismo, está basado en tres conceptos básicos: *Context Attribute*, *Dimension* y *Context Attribute Configuration*. Un *Context Attribute* representa una característica relevante del contexto que puede tomar un valor o un *Context Attribute Value*, desde un conjunto definido. Tipo de proyecto y Tamaño del proyecto son ejemplos de *Context Attributes*. Por ejemplo, para el *Context Attribute* Tipo de Proyecto sus *Context Attribute Values* pueden ser los valores Desarrollo, Mantenimiento o Incidental.

Una *Dimension* es una colección de *Context Attributes* que están relacionados entre sí. Un ejemplo de *Dimension* es Proyecto con los *Context Attributes* Tipo de proyecto y Tamaño del proyecto.

Un *Context* está compuesto por una colección de *Dimensions* y representa el modelo de contexto para una empresa. Una instancia de un modelo de contexto es representada por un *Context Configuration*. Un *Context Configuration* es una colección de *Context Attribute Configurations* que a su vez es una selección de *Context Attribute Values* para cada *Context Attribute*. De esta forma, una *Context Attribute Configuration* relaciona la elección de un único *Context Attribute Value* para cada *Context Attribute*. Por ejemplo, la configuración para un proyecto de mantenimiento donde el *Context Attribute* Tipo de Proyecto con el *Attribute Value* de Mantenimiento.

2.4. Adaptación

Una afirmación aceptada ampliamente dentro de la disciplina de la ingeniería de procesos es que todo proceso de software debe ser adaptado a la situación particular de un proyecto, o bien se torna en un riesgo [5, 7]. Dada la importancia de los procesos a medida en las empresas de software, y al no existir un único proceso que sea igualmente apropiado para todos los proyectos, es importante contar con procesos configurables y flexibles. Una adaptación o *tailoring* está definida como "el acto de ajustar las definiciones y/o particularizar los términos de una descripción general para derivar una descripción aplicable a un ambiente alterno (menos general)" [31]. De forma similar, una adaptación de procesos o *process tailoring*, es el procedimiento en que un proceso de software es configurado para adaptarse a las particularidades de un proyecto [38].

Efectuar una adaptación requiere de un gran conocimiento acerca del modelo de procesos al igual que del contexto del proyecto. Asimismo, es fundamental que la adaptación sea conducida de forma estructurada y disciplinada [25], ya que si es realizada manualmente es intensiva en recursos y propensa a errores. En el marco de ADAPTE, se ha desarrollado una arquitectura basada en *automatic tailoring*. Por otro lado, se propone comparar lo anterior con una adaptación basada en plantillas o *template based tailoring*.

A continuación se exponen las características principales de distintas estrategias de adaptación de procesos: *Automatic tailoring* y *Template-based tailoring*, junto a otras alternativas: *Constructive tailoring*, *Framework based tailoring* y *Self emerging tailoring*.

2.4.1. Automatic tailoring

Es un hecho ampliamente aceptado que la implementación de un proceso de desarrollo de software, que se adecúe al ambiente específico de un proyecto, es un logro en sí mismo al permitir un desarrollo que minimiza los costos y asegura resultados de alta calidad [51]. Esta técnica es conocida como adaptación automática o *automatic tailoring*.

Una estrategia de desarrollo de software que implemente adaptación automática, se basa en la definición y transformación de modelos de procesos de software [75], con el fin de obtener un proceso óptimo con respecto al contexto del proyecto. En particular, ADAPTE ha implementado esta estrategia utilizando MDE mediante la resolución de puntos de variabilidad de un proceso organizacional con respecto al contexto de un proyecto [38, 39, 40]. En esta tesis se pretende comparar experimentalmente los resultados de esta estrategia con un conjunto de procesos predefinidos basados en *template-based tailoring*.

2.4.2. Template-based tailoring

La estrategia de adaptación basada en plantillas o *template-based tailoring*, propone contar con una familia de procesos definidos para diferentes tipos de proyectos según su contexto. En particular, Crystal utiliza *criticality* y *team size* como atributos de contexto para determinar cuál plantilla de proceso será utilizada, e.g., Clear, Yellow, Orange o Diamond, entre otros [23]. Crystal tiene por objetivo ser una metodología de fácil adopción para las empresas, mientras utiliza ideas de XP. La Ilustración 2.9 presenta la matriz de procesos que se generan a partir de estos atributos.

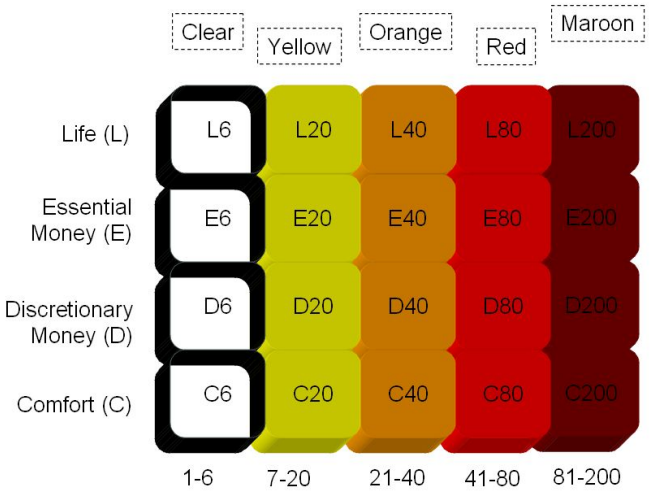


Ilustración 2.9: Una familia de procesos generada por Crystal [23].

Existe la evidencia de que varias de las SSE participantes de ADAPTE poseen un conjunto de tipos de proyectos que efectúan de forma habitual. Esto se debe a que en general dichas empresas se desempeñan en un nicho específico, entonces la cantidad de tipos de proyectos es pequeña y, por lo tanto, el número de procesos requeridos también lo es. Lo anterior refuerza la idea de considerar una adaptación basada en plantillas para comparar con la alternativa propuesta por ADAPTE. Finalmente, la implementación propuesta seleccionará el proceso al comparar el contexto del nuevo proyecto con todos los pertenecientes a la familia de procesos. Se aplicará el proceso correspondiente al contexto más similar al contexto del nuevo proyecto.

2.4.3. Otras Soluciones

Constructive tailoring

A medida que un equipo de trabajo desarrolla una y otra vez productos de similares características, la posibilidad de reutilizar de código se vuelve cada vez más atractiva. En particular, el *constructive tailoring* se basa en componer fragmentos de métodos, integrándolos de manera específica al proyecto.

Situational Method Engineering (SME) promueve la construcción de procesos mediante el ensamblado de fragmentos de procesos desde un repositorio. Esta estrategia también reconoce la necesidad de considerar el contexto para poder obtener el proceso apropiado. Existen diferentes estrategias que siguen SME: *assembly-based* que selecciona métodos existentes, *extension-based* que extiende y adapta patrones existentes, y *paradigm-based* que o bien abstrae un método existente o instancia un metamodelo [70].

Normalmente, la responsabilidad de ejecutar el *constructive tailoring* es compartida entre el jefe de proyectos y el ingeniero de procesos al requerir las decisiones del primero, y la experiencia y el conocimiento del último. Esto impide una separación adecuada de la responsabilidad entre estos roles [4]. Asimismo, el *constructive tailoring* es similar a la adaptación automática al considerar un contexto para obtener el proceso óptimo, pero el primero posee un enfoque *bottom-up* mientras que el último un enfoque *top-down*.

Framework based tailoring

En algunas situaciones donde no existe conocimiento del problema, existen formas de obtener una respuesta de manera inductiva. Asimismo, si no existe información completa para planificar el desarrollo de software, es posible utilizar un *framework* que asista en la búsqueda de método que finalmente permita encontrar una solución [15]. De esta forma, *framework based tailoring* propone métodos generales para enfrentar un proyecto.

El *Rational Unified Process* (RUP) es un *framework* de procesos estructurado ampliamente utilizado en empresas de software [50]. Una de las primeras actividades de RUP es adaptar del proceso [41], pero no existen guías claras de cómo realizar esta tarea. Algunas variantes como el *Process Engineering Process* (PEP) requieren una gran cantidad de recursos, i.e., tiempo y personal, por lo que es una solución apropiada para empresas grandes, pero no para pequeñas o medianas.

Una SSE puede adoptar RUP mediante una reducción sustancial de los roles involucrados, o agrupándolos mediante un análisis detallado [14]. Es posible reconocer la necesidad de adaptar el proceso al proyecto, y por consiguiente facilitar el trabajo del ingeniero de procesos. Si se utilizara una adaptación basada en RUP, se deberían realizar las tareas de reducción y posteriormente de adaptación.

Self emerging tailoring

La estrategia de *self emerging tailoring* ha surgido los últimos años como una propuesta atractiva al considerar el cambio como algo propio del proceso de desarrollo. Las metodologías ágiles y sus prácticas consideran que el contexto cambia en el desarrollo del proyecto, y como consecuencia el proceso. Si bien el objetivo de estas metodologías es agregar valor y producir un producto final, un proyecto puede continuar de forma indefinida si el cambio no

es administrado de forma disciplinada. Debido a lo anterior, las metodologías ágiles disminuyen los riesgos asociados con el cambio de requerimientos, permitiendo la transferencia de conocimiento entre el equipo de trabajo a través del software desarrollado [53]. Scrum y XP son dos metodologías ágiles ampliamente utilizadas que no siguen un proceso estructurado, sino que es creado a medida que el proyecto progresa.

Estas metodologías resultan apropiadas para proyectos pequeños, pero al ser procesos estrictamente no repetibles su análisis se dificulta. Si una empresa quisiera contar con una certificación ISO o una evaluación CMMI, entonces la metodología ágil se vería afectada por estos estándares que usualmente no toman el contexto de la agilidad en cuenta, aunque existen algunas propuestas que sí lo hacen [67].

2.5. Comparación entre modelos

Uno de los desafíos que existen al desarrollar software con una orientación puramente hacia el código, es el constante crecimiento del sistema que está siendo construido. De esta forma, los sistemas de control de versiones permiten al equipo de desarrollo almacenar el código incrementalmente y modificarlo concurrentemente, fomentado el trabajo colaborativo. Los sistemas de control de versiones basados en texto como GIT[7], SVN[8] y CVS[9] obtienen *snapshots* del código al utilizar las operaciones *match*, *diff* y *merge*, i.e., la igualdad, diferencia y unión entre el texto de dos archivos, respectivamente.

En el contexto de MDE, los resultados de utilizar sistemas de control de versiones basados en texto son insatisfactorios, ya que tales sistemas sólo consideran representaciones textuales del modelo, i.e., al realizar cualquier operación existe la posibilidad que la estructura sea destruida perdiendo su información sintáctica. Con esta limitación en mente, se han propuesto estrategias de versionamiento orientadas a modelos que no operan con respecto a las representaciones textuales, sino que consideran directamente el modelo como un grafo [16]. Por un lado el problema de determinar las diferencias entre modelos es intrínsecamente complejo, aunque puede ser separado en tres fases: Cálculo, Representación y Visualización [17]. Para estas fases es posible indicar:

- Cálculo: Es un procedimiento, método o algoritmo que permite comparar dos modelos. La influencia de Alanen y Porres [1] ha intensificado el interés sobre el área de comparación entre modelos, específicamente en lo relativo a diagramas UML [32, 45, 59, 87], aunque también han surgido herramientas que permiten comparar modelos que se encuentren fuera del metamodelo de UML [28, 47, 52, 81].
- Representación: Es necesario encontrar una representación adecuada del modelo para futuras manipulaciones [20]. Por ejemplo, la operación de edición posee una representación es operacional [1, 55], i.e., esta descrita en términos de modificar un modelo

[7]http://git-scm.com/ - Última visita: 18 de Diciembre del 2014
[8]http://subversion.tigris.org/ - Última visita: 18 de Diciembre del 2014
[9]http://cvs.nongnu.org/ - Última visita: 18 de Diciembre del 2014

inicial para obtener el modelo final. Asimismo, una representación adecuada permite combinar automáticamente los resultados con la visualización [59].

- Visualización: Los resultados deben presentarse según una necesidad o alcance específico, al indicar qué información es relevante. Esto es logrado al especificar una sintaxis concreta como texto o diagramas, que expresan la sintaxis abstracta elaborada por la Representación.

El cálculo y representación es central en la comparación entre modelos. En esta tesis, dado que se compararán instancias generadas automáticamente que provienen de metamodelos distintos a UML, es necesario entender los diferentes enfoques y técnicas de comparación entre modelos indicadas en la literatura. Finalmente, se presentan las limitaciones de la herramienta EMF Compare para efectuar la comparación entre modelos y las justificaciones de la implementación propia.

2.5.1. Enfoques de comparación

Según la información disponible sobre los modelos es posible utilizar dos enfoques para operar entre ellos [32]:

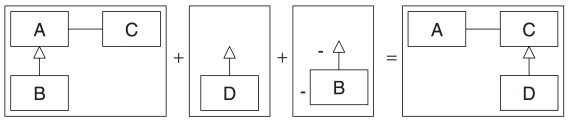
- **Online**: Este enfoque comprende registrar cada cambio que sufren los modelos y cómo fueron realizados; por ejemplo, al contar con funcionalidades en las herramientas de modelado. Sin embargo, cabe la posibilidad de reunir cantidades masivas de datos, ya que la mayoría de los cambios del modelo son pasos intermedios hasta obtener una versión estable. Al registrar los cambios en una herramienta específica, no es posible utilizar modelos provenientes desde otras plataformas.
- **Offline**: Este enfoque es similar a las herramientas de versionamiento basadas en texto, ya que durante la construcción de un modelo se toman *snapshots* que serán comparadas por otra herramienta. Esto permite comparar modelos que han sido creados en diferentes plataformas, mientras sigan un mismo metamodelo.

Las operaciones para realizar la comparación entre modelos son *match*, *diff* y *merge*. En ese sentido, Alanen y Porres definen formalmente el resultado de la operación *diff* como *Delta* (Δ), que es una secuencia de transformaciones que agregan, modifican o eliminan elementos de un modelo. Por otro lado, se define el *merge* como la operación de aplicar un *Delta* a un modelo y así obtener otro. Dados dos modelos M_1 y M_2, basados en un lenguaje en el Meta Object Facility (MOF), se define [1]:

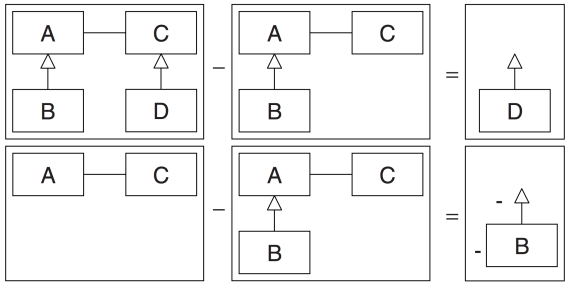
- *Diff* entre dos modelos: $M_2 - M_1 = \Delta$
- *Merge* de un modelo y una diferencia: $M_1 + \Delta = M_2$

Las Ilustraciones 2.10(a) y 2.10(b) muestran ambas operaciones.

Asimismo, las definiciones permiten a las operaciones utilizar un enfoque *online* de comparación, al contar con cada transformación aplicada a los modelos. En de las definiciones mostradas no se presenta el *match* entre modelos, ya que dicha operación está implícita den-



(a) *Merge.*



(b) *Diff.*

Ilustración 2.10: Diagramas para las operaciones de *merge* y *diff*.

tro de las operaciones *diff* y *merge*, i.e., para calcular un *Delta* es necesario identificar los elementos iguales y a partir de ello las transformaciones que componen dicho *Delta*.

Por otro lado, al elegir un enfoque *offline* y definir apropiadamente los modelos, e.g., grafos, estructuras a medida, entonces se cuenta con la información relevante al estado del modelo sin información extra [32], lo que es similar a comparar conjuntos utilizando intersección, diferencia simétrica y la unión entre dos conjuntos.

Al tener acceso a las instancias de los modelos, sean procesos o contextos, el prototipo implementado en esta tesis utiliza el enfoque *offline* para efectuar comparaciones y trata a dichos modelos como estructuras de datos al momento de ser comparados. Estas estructuras estarán basadas en el metamodelo de procesos eSPEM.

2.5.2. *Match* entre modelos

Obtener el *match* entre dos modelos requiere definir la "identidad" de sus elementos constituyentes, y luego encontrar aquellos elementos con "identidades" iguales para considerarlos parte del *match*. La "identidad" es calculada mediante una función de *match*, sin embargo, las características particulares de un elemento de modelo dentro de la función de *match* dependen de los enfoques, escenarios y objetivos al realizar la comparación entre modelos [16].

Ambos predecesores del *match* entre modelos fueron el *schema matching* [69] y el *ontology matching* [85], utilizados en bases de datos y en representación de información, respectivamente. La Ilustración 2.11 muestra la clasificación propuesta por Rham y Bernstein, que consolida la estructura y terminología utilizada en ese momento al clasificar las estrategias disponibles, según cuáles eran los algoritmos utilizados. Inicialmente estos *matchers* se distinguen entre individuales y combinados.

Como su nombre lo indica, los *matchers* individuales utilizan sólo un criterio en su función de *match*, más aún, se considera la siguiente clasificación ortogonal [69]:

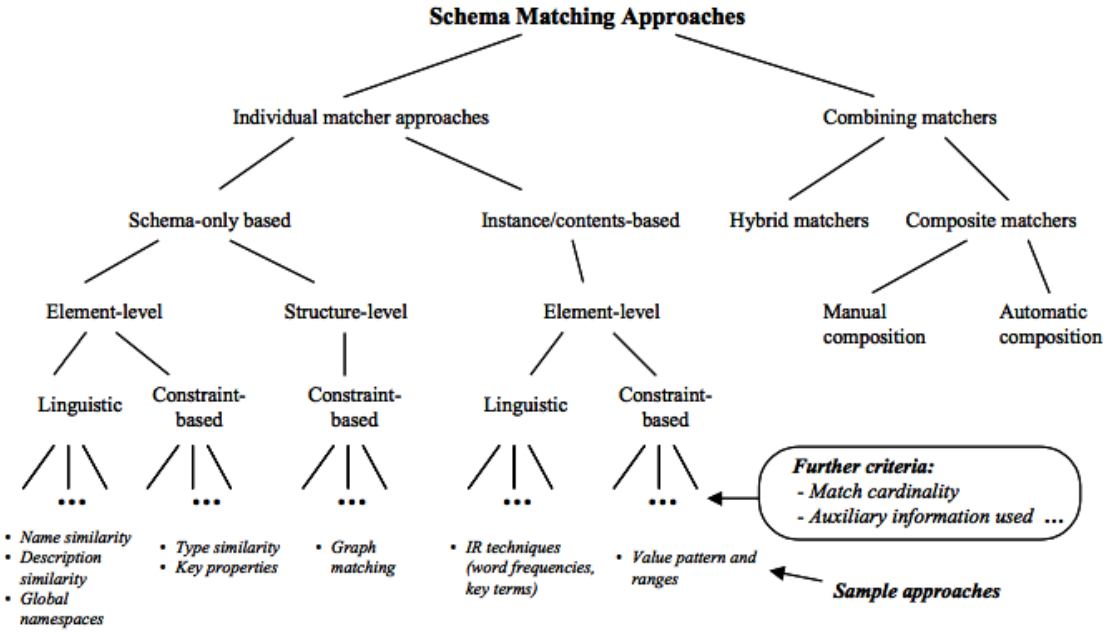


Ilustración 2.11: Clasificación de *matchers* - [69]

- *Instance vs schema*: Las estrategias se dividen según qué tipo de información que incorporan, ya sea los datos de la instancia o la información del *schema*.
- *Element vs structure matching*: A su vez, las estrategias basadas en el *schema* se clasifican según si el *match* puede ser realizado según elementos individuales del *schema* como atributos, o bien si es realizado según combinaciones de estos elementos como estructuras mismas del *schema*.
- *Language vs constraint*: Un *matcher* puede utilizar una estrategia basada en lingüística, e.g., nombres o descripciones textuales, o una estrategia basada en restricciones, e.g., propiedades de llave única o tipos de datos de los elementos del *schema*.
- *Matching cardinality*: El resultado de un *match* puede relacionar uno o más elementos entre dos *schemas*, produciendo cuatro cardinalidades: 1:1, 1:n, n:1, n:m. Asimismo, pueden existir diferentes cardinalidades para el *match* al nivel de instancia.
- *Auxiliary information*: La mayoría de los *matchers* no sólo se basan en los *schemas* entregados como input, sino también en información auxiliar como diccionarios, *schemas* globales, decisiones previas de *matching* e *input* del usuario.

En el caso de los *matchers* combinados se clasifican de la siguiente forma [69]:

- *Hybrid vs Composite*: Las estrategias combinadas se caracterizan según si son híbridas, que directamente combinan múltiples criterios de *matching* individual para determinar los candidatos; o combinados, que combinan los resultados de las ejecuciones independientes de otros *matchers* individuales.
- *Manual vs Automatic*: Combinar automáticamente los *matchers* puede reducir el número de interacciones de un usuario, pero es difícil lograr una solución genérica que sea adaptable a diferentes dominios. De forma alternativa, un usuario puede seleccionar los *matchers* a ser ejecutados, el orden y cómo combinar sus resultados; la forma manual es más fácil de implementar y otorga más control al usuario.

Las clasificaciones y terminologías anteriores son utilizadas por Kolovos et al. para proponer una categorización específica a las estrategias de *match* entre modelos. Es posible distinguir [48]:

- *Static identity-based matching*: Se basa en atributos libres de semántica asociados a cada elemento del modelo. En general, trata al problema como una identidad de verdadero o falso, i.e., dos elementos son un *match* o no lo son.
- *Signature-based matching*: Compara los elementos del modelo basándose en una combinación de sus valores de características, i.e., su *signature*. Estas características dependen fuertemente del lenguaje de modelado. En general, trata al problema como una identidad de verdadero o falso, i.e., dos elementos son un *match* o no lo son.
- *Similarity-based matching*: Calcula una medida de similitud agregada entre dos elementos del modelo basándose en los valores de sus características. Como no todos los valores de características de un elemento son significantes, se les asignan pesos para representar esta importancia.
- *Custom language-specific matching*: Permite a sus usuarios específicamente decidir las reglas y el orden en que se aplican, según sus criterios de acuerdo con la semántica del lenguaje de modelado.

A continuación se presentan las estrategias mencionadas en el dominio de *matching* entre modelos. Muchas de las estrategias en este dominio están integradas en versionamiento de modelos, sin embargo, se revisarán sus capacidades sólo con respecto al *matching*, además de discutir su viabilidad con respecto a la implementación de la solución propia. Ahora bien, no se revisarán estrategias para las operaciones *diff* y *merge*, puesto que únicamente se quiere obtener el *match* entre modelos pertenecientes en la capa M1 (ver sección 2.1) y la calidad de la operación de *match* es crucial para que las operaciones subsecuentes sean precisas.

Static Identity-Based Matching

Esta estrategia utiliza atributos libres de semántica, i.e., atributos tipográficos o lingüísticos y, por lo tanto, es posible establecer similitudes según el nombre o identificador de los elementos que componen a los modelos. Las formas de *static identity-based matching* [58] son el *matching* tipográfico y el *matching* lingüístico.

El *matching* tipográfico tiene por restricción que cada elemento tiene un identificador único que es asignado al momento de su creación, e.g., UUIDs. Para comparar un par de elementos se verifica que los identificadores sean iguales, asimismo es posible aplicar un algoritmo *N-gram* [54] a los identificadores, asignándoles un umbral entre cero y uno, donde cero representa ninguna similitud y uno igualdad. Su principal ventaja es que requiere un esfuerzo de configuración bajo, ya que sólo se debe establecer un umbral de similitud.

En el caso de aplicar esta estrategia a modelos, la restricción de que todos los elementos posean identificadores únicos es necesaria. Por ejemplo para los modelos en la Ilustración 2.12,

si son creados con herramientas distintas, entonces sus identificadores serán distintos y el uso de otra estrategia de *matching* es preferible. En el caso de que fueran creados dentro de la misma herramienta, las modificaciones entre ambos modelos pueden ser representados como cambios de los elementos, i.e., al detectar que las clases Cliente y Cuenta fueron colocadas dentro del nuevo *package* Usuario, o bien fueron borrados y luego agregados en el nuevo *package* Usuario.

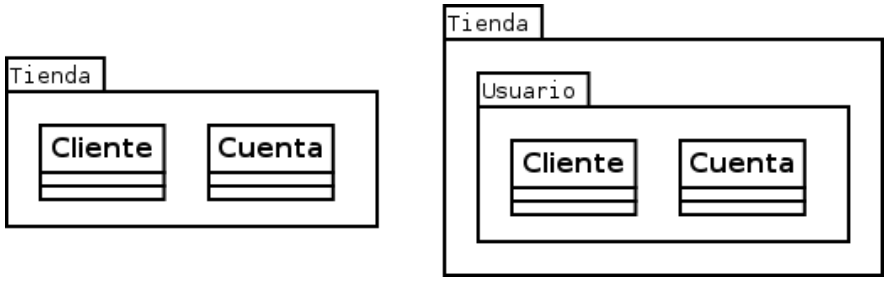


Ilustración 2.12: El *matching* tipográfico no siempre permite identificar cambios estructurales.

Por otro lado, el *matching* lingüístico [65] considera las reglas y restricciones que definen la correctitud estructural de un modelo [19], e.g., una clase debe tener un único nombre, una relación tiene un determinado origen y destino. Al cuantificar la similitud entre los conceptos según correlaciones lingüísticas que están representadas por un valor entre cada par de palabras con respecto a la relación elegida, es posible establecer un orden entre las palabras. Lo anterior permite crear operadores sobre modelos, definir restricciones sobre ellos, transformaciones y administrar su evolución. Epsilon[10] es una familia de lenguajes y herramientas basados en el trabajo realizado por Rose et al. [72]. Epsilon otorga una gran variedad de algoritmos de comparación y *matching* de modelos que pueden ser configurados. La Tabla 2.2 muestra ejemplos de estas relaciones y sus valores correspondientes.

Relación	Diccionario de pares de palabras
is-a	(Metaclass, Class)
has-a	(Class, Attribute), (Class, Operation)
belongs-to	(Cliente, Usuario), (Cuenta, Usuario)
comforms-to	(Model, Metamodel), (Metamodel, Metametamodel)

Tabla 2.2: Ejemplos de relaciones para comparaciones lingüísticas.

En esta tesis es posible utilizar un *matching* tipográfico para comparar entre las instancias de un modelo de contextos o contexto organizacional, ya que las instancias se generan desde la herramienta *Context Model* [63] y poseen identificadores únicos.

Signature-Based Matching

Esta estrategia compara los elementos basándose en una *signature*, que es una combinación entre valores o características pertenecientes a los respectivos elementos del modelo que son

[10]http://www.eclipse.org/epsilon/ - Última visita: 18 de Diciembre del 2014

específicos al lenguaje de modelado. Asimismo, una *signature type* es el conjunto de elementos usado para determinar una *signature* [71]. La Ilustración 2.13 muestra los modelos A y B, que contienen la clase Cliente con los atributos Nombre y Dirección, y clase Cliente con el atributo Nombre, la operación Update y una asociación a la clase Cuenta, respectivamente.

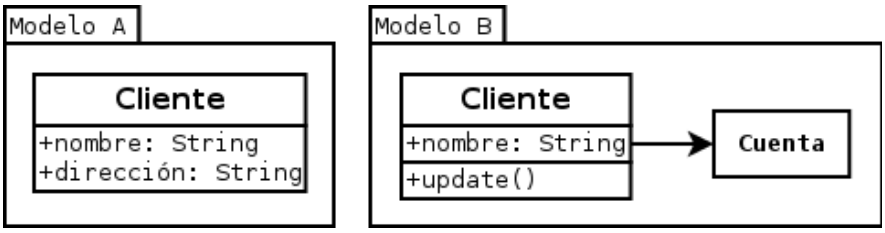


Ilustración 2.13: Modelos A y B con sus clases correspondientes.

En MDE, para comparar los modelos A y B de la Ilustración 2.13, es necesario definir una *signature type* que contemple la estructura del metamodelo de dichas instancias, i.e., UML. En este caso, se define la *signature type* como una clase que posee un nombre y está compuesta de uno o más atributos y asociaciones. A su vez, los atributos se componen de un nombre y un tipo, e.g., Integer, String, y las asociaciones poseen un origen y un destino. La Tabla 2.3 muestra la *signature type* descrita y las *signatures* para los modelos A y B:

Signature type	{clase (nombre, {atributo (nombre, tipo)}$^+$, {asociación (origen, destino)}$^+$) }
Signature A	{Cliente {(nombre, String), (dirección, String)} }
Signature B	{Cliente {(nombre, String), Cuenta {(número, Integer), (Cliente, Cuenta)} }

Tabla 2.3: *Signature type* y *signatures* para los modelos A y B.

Al igual que la estrategia *static identity-based*, el *signature-based matching* resultado de comparar es si ambos elementos son iguales o no. Para lo anterior se requiere definir cuáles elementos de la *signature type* serán considerados y luego comparar las *signatures* [71]. Por ejemplo, si las *signatures* A y B son comparadas con respecto a toda su *signature type*, i.e., al considerar los nombres de las clases, atributos y asociaciones, son distintas. Sin embargo, si ambas *signatures* son comparadas con respecto al nombre de una de sus clases, el *match* existe debido a la clase Cliente.

En esta tesis se descarta el uso del *signature-based matching*, debido a que los modelos de contextos o de procesos son particulares para cada empresa, así como sus instancias. Entonces definir una *signature type* es una tarea específica en cada empresa y se quiere lograr una solución más general.

Similarity-Based Matching

Esta estrategia plantea establecer un *match* estructural entre modelos al considerarlos como *typed attributed graphs* (TAG), i.e., una colección de nodos y arcos con atributos [30].

Cada elemento posee un conjunto de atributos que permite identificarlo como único, e.g., un nodo es una 4-tupla formada por (nombre, tipo, especie, atributos) mientras que un arco es una 5-tupla de (nombre, tipo, especie, origen, destino). Entonces, para lograr el *match* entre modelos se utiliza *signature based matching* para generar dos criterios de evaluación de elementos, según la composición de sus atributos [52]:

- **Node Signature Matching**: Sean los modelos M_1 y M_2 con sus respectivos nodos v_{m1} y v_{m2}. Los nodos son similares si sus *signatures* son textualmente equivalentes.
- **Edge Signature Matching**: Sean los modelos M_1 y M_2 con sus respectivos arcos e_{m1} y e_{m2}. Los arcos son similares si sus *signatures* son textualmente equivalentes.

Por ejemplo en la Ilustración 2.14, la clase Cliente del modelo A posee dos candidatos en el modelo B. Si bien ninguno es textualmente equivalente, existen características que son más relevantes que otros, e.g., el nombre de la clase es mucho más importante que su atributo *abstract*, sus atributos u operaciones. Según *node signature matching*, las clases Cliente de ambos modelos son más similares que las clases Cliente y Cuenta. Por otro lado, ambos modelos cumplen con *edge signature matching* al no poseer ningún arco y, por lo tanto, son equivalentes bajo este criterio.

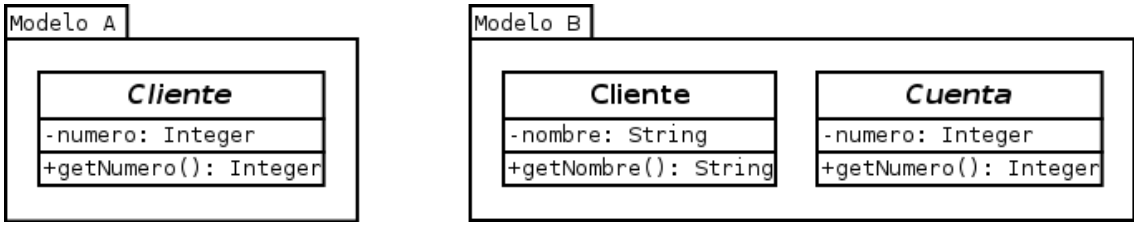


Ilustración 2.14: Según *node signature matching*, la clase Cliente del modelo B es el mejor candidato para la clase Cliente del modelo A.

Ambas definiciones contribuyen para evaluar los elementos candidatos entre los modelos. Para establecer el *matching* estructural se define *Edge Similarity* [52]:

- **Edge Similarity**: Sean dos modelos M_1 y M_2 con sus respectivos nodos v_{m1} y v_{m2}, donde v_{m2} es candidato para v_{m1}. El *edge similarity* de v_{m2} con respecto a v_{m1} es la cantidad de arcos que conectan a v_{m2}, tal que los arcos que conectan con v_{m1} cumplan con *edge signature matching*.

La Ilustración 2.15 muestra que para las clases Cliente y Cuenta es posible asumir que logran un *node signature matching* y, por lo tanto, cada una es candidata para la igualdad, pero los modelos son estructuralmente distintos debido a que el arco presente no cumple con *edge signature matching* ni con *edge similarity*.

En MDE, esta estrategia es posible ajustarla a diferentes escenarios dependiendo del metamodelo a considerar. Aunque en el caso particular de esta tesis, no existen diferencias estructurales entre dos instancias provenientes un mismo modelo de contextos, sólo los *Context Attribute Values* de sus *Context Attributes*. Asimismo para las instancias de un modelo de procesos, obtener las diferencias estructurales entre las instancias no es un objetivo, sino

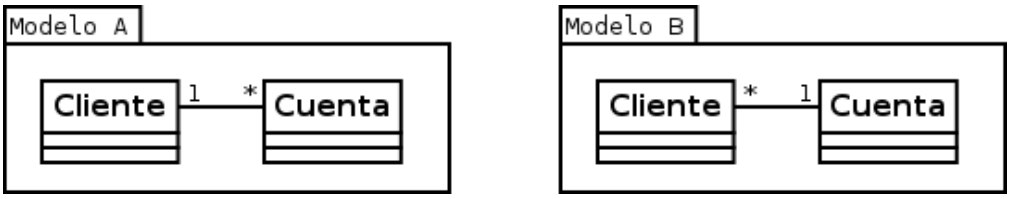


Ilustración 2.15: Según *edge signature matching* los modelos son distintos.

que la cantidad de elementos extras y faltantes entre dichas instancias. Las características de los elementos del modelo al no ser igualmente importantes para efectuar el *matching*, pueden ser configuradas en términos de pesos a los elementos relevantes en la comparación. Esta noción es muy útil para definir la comparación entre instancias de un modelo de contexto.

Custom Language-Specific Matching

Finalmente, *custom language-specific matching* permite a sus usuarios definir reglas específicas que permitan calcular el *match* y considerar la semántica del metamodelo, e.g., UML, como el recurso principal para implementar operaciones. Dada la particularidad de los metamodelos eSPEM y SPCM, además de la representación y flexibilidad necesaria para el manejo de sus instancias, esta estrategia será utilizada en el desarrollo de la solución.

Al contrario de algunas herramientas implementadas específicamente para UML, no se abordará la visualización de los resultados de la comparación y, por lo tanto, la solución no necesita considerar la posición de los elementos en el modelo [58], generando una solución que se apegue a las particularidades de los metamodelos.

Con relación a esta tesis, como se ha mencionado anteriormente, se utilizará *static identity-based matching* para comparar entre las instancias de los contextos organizacionales debido a sus estructuras. Por otro lado, la ventaja principal de esta estrategia es que no es necesario considerar los identificadores únicos de los elementos en un archivo XMI, e.g., UUIDs, para efectuar una comparación, lo que permite comparar dos instancias de un mismo proceso organizacional, basándose en el metamodelo eSPEM. Lo anterior se efectúa al definir los *Method Content Elements* y sus usos en una *Activity*, y generar una estructura de datos que represente al proceso. Finalmente, el *match* entre dos procesos se realiza al comparar las estructuras de datos correspondientes, elemento a elemento. Asimismo, se tendrá mayor control de cómo son implementadas las comparaciones.

2.5.3. Herramientas

La sección anterior presenta los conceptos y técnicas fundamentales que permiten generar una base para establecer herramientas que permitan aplicar tales conceptos. Durante los

últimos años han surgido una cantidad notable de propuestas para efectuar el *match* entre modelos; desde estrategias basadas en un lenguaje particular hasta estrategias híbridas y compuestas, donde algunas son adaptables y algunas no lo son. La mayoría de ellas operan a un nivel estructural sin tomar en cuenta la importancia del contexto de cada elemento a operar, salvo el caso de Epsilon Comparison Language (ECL) que de forma explícita permite considerar conocimiento externo además de semántica, e.g., la semántica de UML dicta de que no pueden existir dos operaciones en la misma clase, con el mismo nombre y los mismos parámetros. A continuación se presenta la herramienta EMF Compare, además de una discusión acerca de sus capacidades en el contexto de esta tesis.

EMF Compare

Eclipse Modeling Framework Compare[11] (EMF Compare) [17] es parte del proyecto *open source* Eclipse Modeling Framework Technology[12], que permite comparar y efectuar *merge* entre modelos de forma genérica. Lo anterior se traduce en un framework estable, eficiente y extensible para la comparación entre modelos.

En EMF Compare, la comparación entre dos modelos consiste en dos fases: la fase de *matching* y la fase de *differencing*. Para la fase de *matching* es posible utilizar *similarity-based matching* al calcular cuatro métricas que son combinadas para obtener la medida de similitud: el nombre del elemento, su contenido, su tipo y las relaciones con otros elementos; basándose fuertemente en la estructura de grafos al comparar ambos modelos. También es posible utilizar una comparación basada en *static identity-based matching*. Sin embargo, EMF Compare sólo permite una de ellas a la vez, i.e., sólo *static identity-based matching* o sólo *similarity-based matching*, pero no combinadas.

En la fase de *differencing*, es posible ver las diferencias entre los archivos de texto de ambos modelos o nuevamente utilizar la estructura de grafos para visualizar dichas diferencias. Por ejemplo, la Ilustración 2.16 muestra un extracto de la comparación entre dos procesos—representados por los archivos ProcessA.xmi y ProcessD.xmi—para la actividad llamada Análisis de Requerimientos de la empresa Mobius. Esta herramienta resalta las diferencias entre ambos procesos: comparado con el proceso ProcessA.xmi, el ProcessD.xmi no posee la tarea tsk_ui_design.

Ahora bien, la Ilustración 2.17 muestra la comparación para la actividad Inicialización que está compuesta de las 2 tareas (tsk_recieve_client_request y tsk_create_wiki_space) y 4 actividades (cp_requirements_analysis, cp_project_planning, cp_commercial_agreement, cp_launch). Sin embargo, la comparación entre los archivos ProcessA.xmi y ProcessD.xmi, no arroja diferencia alguna aunque la actividad Análisis de Requerimientos posea diferencias. Esto constituye la limitante en el uso de EMF Compare para esta tesis, puesto que no permite obtener de forma inmediata la diferencia entre dos procesos, y efectuar de forma manual el

[11] http://www.eclipse.org/emf/compare/ - Última visita: 18 de Diciembre del 2014
[12] http://eclipse.org/modeling/emft/ - Última visita: 18 de Diciembre del 2014

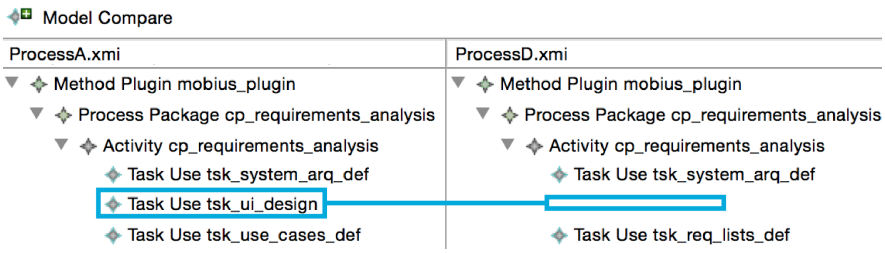


Ilustración 2.16: Mobius: Diferencias para la actividad Análisis de Requerimientos.

cálculo es propenso a errores. Es por lo anterior que se decidió generar un prototipo propio que permitiera comparar dos procesos de manera sistemática.

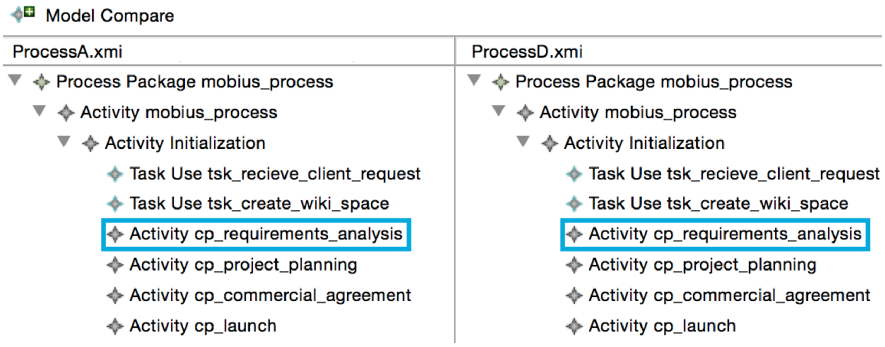


Ilustración 2.17: Mobius: Diferencias para la actividad Inicialización.

Capítulo 3

Problema u Oportunidad

En el Capítulo 1: Introducción, el tema a abordar en la tesis fue explicado en rasgos generales, al igual que las implicancias de una solución. Este capítulo abarcará en detalle ambos elementos, además de proveer justificaciones acerca de las suposiciones y decisiones que enmarcan el planteamiento del problema.

3.1. Planteamiento del problema

El problema que motiva el desarrollo de esta tesis se enmarca dentro de ADAPTE, proyecto Fondef financiado por Conicyt, Chile. Para las empresas de software, formalizar un proceso organizacional de software les permite evaluar su nivel de madurez, además de planificar y alcanzar los niveles de calidad, costos y plazos que mejoran su productividad y competitividad. Esto es una premisa ampliamente aceptada en la industria: a mayor formalidad de un proceso, mayor será la calidad de que los productos de software desarrollados y la probabilidad de que cumplan, e incluso excedan las expectativas del cliente [18]. En este sentido, un grupo acotado de empresas ha definido su proceso organizacional usando técnicas e ideas provenientes de las prácticas más populares y de su propia experiencia, pero es un artefacto que se utiliza sin realizar una adaptación o simplemente no se utiliza.

Adaptar e instanciar un proceso para un proyecto requiere conocimientos específicos de la disciplina y de la empresa, i.e., diseño de procesos, caracterización de proyectos, y caracterización del entorno de desarrollo. Entonces para el común de los proyectos en una SSE rara vez se realiza una adaptación, más aún cuando requiere un alto conocimiento sobre de los estándares existentes —CMMI [22, 88] e ISO—y su aplicación. ADAPTE, con el objetivo de disminuir esta barrera, ha creado e implementado una herramienta metodológica y tecnológica que apoye el mejoramiento de procesos las SSE, utilizando un enfoque de Software Product Lines (SPrL), para evitar la proliferación de errores y disminuir el tiempo para efectuar una adaptación. Dado lo anterior, ADAPTE fue ejecutado en 4 fases enunciadas a continuación:

1. Definición o ajuste formal de los procesos.

2. Especificación de variabilidad en los procesos organizacionales.
3. Caracterización y representación del contexto de un proyecto.
4. Diseño de una adaptación de procesos organizacionales a contextos de un proyecto para su ejecución.

Finalizadas de forma satisfactoria cada una de las fases, ADAPTE pudo proveer el conocimiento requerido (diseño de procesos, caracterización de proyectos, caracterización del entorno de desarrollo) para generar una estrategia de adaptación basada en MDE que automáticamente adapta un proceso organizacional al contexto de un proyecto, produciendo un proceso adaptado aplicable en dicho proyecto. Ahora bien, en la última fase se plantea el problema que motiva el desarrollo de esta tesis: *"¿cuál es el beneficio para una empresa de software de implementar una estrategia de adaptación automática de procesos?"*

Responder la pregunta anterior comprende comparar las estrategias de adaptación automática de ADAPTE y una basada en plantillas con respecto al beneficio que ofrecen. Esta decisión se sustenta en la experiencia del equipo de ADAPTE al apoyar y guiar las instancias de formalización de los procesos organizacionales en las SSE. En particular, las empresas Rhiscom y Mobius adaptaban o seleccionaban el proceso basándose en la experiencia de personas involucradas en el desarrollo de sus proyectos expresado en tres formas de adaptación: (1) manual sólo al comienzo del proyecto, (2) manual al comienzo y durante el transcurso del proyecto y (3) basada en plantillas. Como los enfoques manuales no son repetibles o comparables dentro de una misma empresa, incluso en proyectos similares, se representará el funcionamiento de las empresas mediante una adaptación basada en plantillas llamada desde ahora Procesos predefinidos.

Dada la necesidad de establecer cuáles elementos permiten comparar las estrategias de adaptación, se ha realizado un piloto conceptual basado en el *Goal/Question/Metric Paradigm* (GQM) [6, 8]. El GQM plantea definir un conjunto de metas que son refinadas mediante una o más preguntas las que a su vez son cuantificables mediante métricas. El piloto se ejecutó utilizando el proceso organizacional de Mobius y gracias a los aprendizajes del mismo fue posible generar una solución que se especificará en el Capítulo 4: Diseño de la solución. Los pasos del piloto conceptual fueron los siguientes:

- Se definieron los contextos correspondientes a los tipos de proyectos más recurrentes en Mobius.
- Se realizó la adaptación de forma manual sólo para la fase de Análisis de Requerimientos.
- Se compararon los productos de la adaptación manual.
- Se definieron tres contextos al azar, los que se compararon con los contextos previamente definidos.

Con lo anterior, fue posible plantear las siguientes metas, preguntas y métricas:

Meta 1: Comparar dos estrategias de adaptación de procesos según el beneficio que proveen en los proyectos de una empresa.

- *Pregunta* 1.1: ¿Cómo se mide la *productividad* de un proceso adaptado automáticamente

en un proyecto?

Métrica 1.1: Tareas extras para el contexto i:
tareas en el match de PA_{i} y PP_{i} - tareas del proceso predefinido PP_{i}

- *Pregunta* 1.2: ¿Cómo se mide la *calidad* de un proceso adaptado automáticamente en un proyecto?

  Métrica 1.2: Tareas faltantes para el contexto i:
  tareas en el match de PA_{i} y PP_{i} - tareas del proceso automáticamente adaptado PA_{i}

Meta 2: Para un proyecto, seleccionar un proceso desde una familia de procesos.

- *Pregunta* 2.1: ¿Cómo seleccionar el mejor proceso de software para un proyecto?
  Métrica 2.1: Similitud entre dos contextos *i* y *j*:

$$\frac{\textit{Context attribute values} \text{ iguales para cada } \textit{Context attribute}}{\text{número de } \textit{Context attributes}}$$

Al no existir una manera definida para comparar el beneficio de una estrategia de adaptación de procesos, se decide cuantificar dicho beneficio al comparar el número de tareas extras y tareas faltantes entre dos procesos de software—uno obtenido desde la adaptación automática y otro seleccionado entre los procesos predefinidos—considerando un mismo conjunto de contextos. Por otro lado, como no se ha mostrado la manera de seleccionar el mejor proceso desde los procesos predefinidos para un contexto en particular, se define la similitud que es un valor entre cero y uno y que a mayor valor indica una mayor cantidad de *Context attribute values* iguales.

El foco de las preguntas y métricas es en analizar el desempeño de las estrategias en un ambiente donde no se han ejecutado proyectos, basándose en consideraciones prácticas como la experiencia de las personas relacionadas con la definición del proceso, la disponibilidad de los datos en ese momento y el alcance de esta investigación. En ese sentido, las métricas son independientes del uso de las estrategias y pueden ser utilizadas antes de la ejecución de los procesos. Ahora bien, vale la pena enumerar los logros parciales y aprendizajes producto del piloto conceptual:

- Con respecto a la adaptación automática:
  1. Sólo las tareas y actividades son evaluadas al efectuar una adaptación automática.
  2. Para un usuario final, sólo es necesario definir un contexto para obtener el proceso adaptado; la lógica de la adaptación no es visible. Se utilizará la misma estructura para los procesos predefinidos.
- Con respecto a los procesos predefinidos:
  1. Generar la familia de procesos es una tarea que requiere un conocimiento profundo sobre la empresa.
  2. Cada proceso (PP_{i}) predefinido está representado mediante un contexto *i*. Si se realizara una adaptación automática con un contexto *1*, el proceso (PA_1) debe ser el resultado.

3. En un principio, la manera de seleccionar un proceso desde la familia de procesos estaba indeterminada. Se definió el concepto de similitud para este fin.

- Con respecto a la comparación entre procesos:
  1. Comparar dos procesos manualmente es una tarea que implica un gran esfuerzo y de todas formas es propensa a errores. Es necesario realizarlo de una manera sistemática para poder obtener resultados válidos.
  2. Fue posible medir la cantidad de tareas faltantes al comparar dos procesos que son las tareas que no están presentes en el proceso predefinido, pero sí en el proceso adaptado automáticamente.
  3. Fue posible medir la cantidad de tareas extras al comparar dos procesos que son las tareas que están presentes en el proceso predefinido, pero no en el proceso adaptado automáticamente.

3.2. Relevancia de la solución

Actualmente no existe un método para decidir cuál estrategia de adaptación de procesos de software utilizar en el contexto de las SSE. Entonces, la relevancia de esta tesis yace en evaluar la pérdida (o ganancia) de aplicar la adaptación automática propuesta por ADAPTE o los procesos predefinidos. Hoy en día, seleccionar una estrategia de adaptación es una decisión subjetiva, tomada desde un punto de vista estratégico o basada en la experiencia de las personas claves dentro de la organización, e.g., ingenieros de procesos, jefes de proyectos. Debido a lo anterior, es útil contar con una evaluación cuantitativa que involucre aspectos concretos de la organización.

Desde el punto de vista académico esta tesis explora un área inmadura, como lo indica la revisión literaria sobre *Software Process Lines* de Carvalho et al. [27], que considera un total de 40 artículos publicados entre los años 1996 y 2013; sólo dos (5 %) entre 1996 y 2003, mientras que 26 (65 %) entre los años 2010 y 2013. Cabe considerar que ese 5 % corresponde al trabajo de Sutton y Osterweil [80] y de Gomaa et al. [33], donde fueron sentadas las ideas bases para *process families* además de su perspectiva acerca de la reutilización del software. Los artículos incluidos consideran que las empresas desean certificar su proceso de desarrollo de software de forma rigurosa a través de estándares y modelos, e.g., CMMI-DEV, MR-MPS-SW, ISO/IEC 12207, y utilizan un enfoque de SPrL para modelar dichos procesos. Asimismo, se identifican diferentes modelos, estándares, frameworks de procesos y ciclos de vida, que utilizan los conceptos de SPrL. Sin embargo, la revisión critica la ausencia en la definición del alcance, consistencia, aplicabilidad y validación de los estándares, puesto que fueron modelados como un ejemplo de aplicación. Lo anterior indica la ausencia de artículos que apliquen conceptos de SPrL a modelos de procesos y estándares utilizados por la comunidad de ingeniería de software, y menos aún, una evaluación objetiva sobre la conveniencia de adaptarlos.

La existencia una extensa investigación sobre las estrategias de comparación, diferencia y unión de modelos—como objetivos en sí mismas—refleja un área cada vez más madura con

respecto a los lenguajes, procedimientos de comparación y las herramientas involucradas; lo que está reflejado en el extenso trabajo mostrado en el capítulo anterior. Por otro lado, la factibilidad y uso de herramientas está ampliamente documentado [28, 37, 47, 51, 52, 63, 76, 82, 83], sin embargo, no se han utilizado estas operaciones como medios para evaluar estrategias de adaptación de procesos, lo que otorga una validación y un enfoque novedoso al área.

Finalmente, la revisión indica que existe una cantidad limitada de artículos que efectúan comparaciones para indicar las ventajas o desventajas de su estrategia con respecto al trabajo relacionado reportado. Asimismo, la falta de madurez en el área está reflejada en la ausencia de una taxonomía y una evaluación de calidad de las estrategias propuestas, además la ausencia de mejoras en términos de una validación empírica. Carvalho et al. [27] también indica que dado el incremento en el número de publicaciones, que muestran evidencia tanto en la academia como en la industria, que el paradigma de SPrL es factible y beneficioso de aplicar en un contexto real; pero es necesario contar con métodos más rigurosos como un *case study*. Esto último se alinea directamente con el problema presentado en esta tesis.

Capítulo 4

Diseño de la solución

Para responder al problema planteado *"¿cuál es el beneficio para una empresa de software de implementar una estrategia de adaptación automática de procesos?"*, es necesario considerar en detalle la adaptación automática propuesta por ADAPTE y posteriormente describir el procedimiento de creación y selección de los procesos predefinidos. Ya establecidas ambas estrategias, es posible comparar los procesos de software—según sus elementos de procesos—obtenidos por adaptación o selección. El objetivo de este capítulo es presentar el diseño de la solución para el problema planteado.

4.1. Adaptación automática

Como se ha indicado anteriormente, la adaptación automática consiste en definir y posteriormente transformar un modelo de procesos de software con respecto a las características de un proyecto específico, para obtener un proceso adaptado que es óptimo. La Ilustración 4.1 muestra los elementos involucrados: un proceso organizacional y el contexto de un proyecto, que conforman con eSPEM y SPCM respectivamente, y una transformación de modelos implementada mediante ATL (ATLAS Transformation Language) [39].

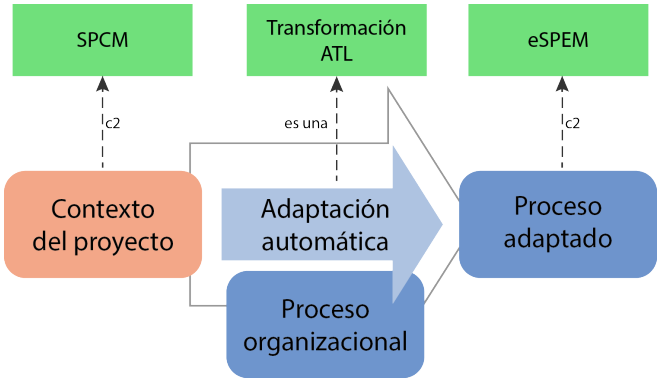


Ilustración 4.1: Estrategia de adaptación automática de procesos utilizada por ADAPTE.

ADAPTE al contar con el metaproceso CASPER [36], cuenta con una serie de prácticas para el desarrollo y aplicación de procesos de software. Sin embargo, hacer efectivas dichas prácticas no es una tarea sencilla para una SSE. Por una parte, el contexto organizacional es considerado un requerimiento en el subproceso de *process domain engineering*, ya que debe estar determinado antes de diseñar o comparar los procesos. Para generar el contexto organizacional de una SSE, se recurre a todas las fuentes de información, incluyendo, pero no limitado a: proyectos pasados, expertos, adaptaciones previas, tipos de proyectos identificados, plantillas. Luego, el contexto organizacional es formalmente especificado como una instancia de SPCM al definir sus *Context Attributes* y *Context Attribute Values*, organizados en *Dimensions*. La Ilustración 4.2 muestra la herramienta Context Model [63] utilizada para este propósito.

Ilustración 4.2: Ejemplo de uso de la herramienta Context Model [63].

Definido el contexto organizacional, es posible configurar el contexto de un proyecto utilizando la herramienta Concrete Context [63]. Esta herramienta consume la salida de la herramienta Context Model y permite configurar los *Context Attributes* según los *Context Attribute Values* previamente definidos. La Ilustración 4.3 muestra su uso.

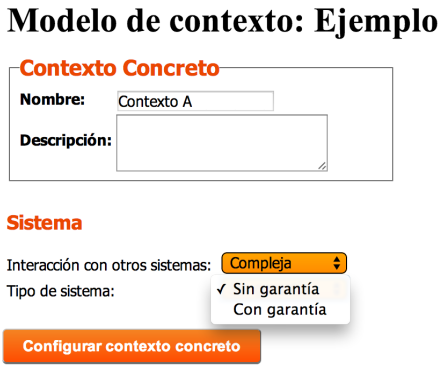


Ilustración 4.3: Ejemplo de uso de la herramienta Concrete Context [63].

Siguiendo el metaproceso, ADAPTE forma el SPEG mientras que la empresa actúa como el PT. Utilizando eSPEM y la plataforma EPF Composer es posible generar el proceso organizacional con variabilidades, además de un diagrama de actividades; artefacto que permite visualizar a los *stakeholders* la formalización del proceso de software. La Ilustración 4.4

muestra el diagrama de actividades de Análisis de Requerimientos para la empresa Mobius, donde se visualizan los elementos mencionados: las tareas representadas mediante el ícono de *Task Use* (definido en SPEM 2.0) con las reglas de variabilidad asociadas para cada tarea, representadas por anotaciones en amarillo. Las anotaciones asociadas directamente a una tarea denotan que dicha tarea es opcional, mientras que si está asociada una condición significa que las tareas involucradas son alternativas.

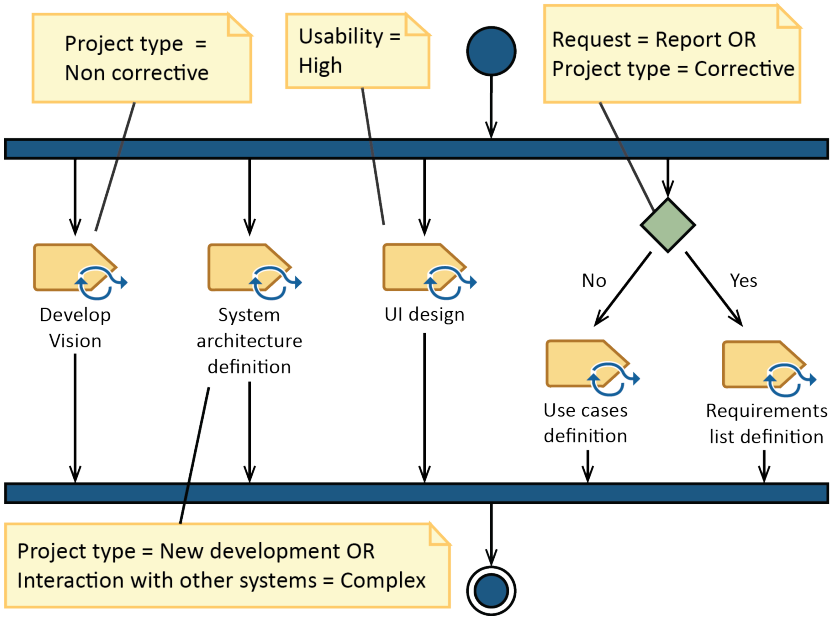


Ilustración 4.4: Mobius: Diagrama de actividades para Análisis de Requerimientos.

Formalizados el proceso y el contexto organizacional, junto a la posibilidad de generar contextos para cada proyecto, sólo es necesario definir qué es y cómo funciona una transformación de modelos para la adaptación automática. Como se ha mencionado anteriormente, el desarrollo de un proyecto que utiliza MDE contempla el uso de una serie de transformaciones sobre modelos. Cabe recordar que una transformación es la refinación progresiva de un modelo, donde usualmente es disminuido su nivel de abstracción, hasta volverlo un artefacto adecuado para una tarea específica (Sección 2.1). En el caso de la adaptación automática se utiliza ATL (ATLAS Transformation Language), el cuál es un lenguaje *domain-specific* para especificar transformaciones de modelo a modelo (M2M) [42].

Una transformación en ATL está compuesta por un conjunto de reglas, que tienen por objetivo adaptar el proceso organizacional a los *Context Attribute Values* de un contexto. A su vez, las reglas están formadas de un *source pattern* y un *target pattern*. El primero especifica un conjunto de *source types* proveniente del metamodelo y una expresión booleana, y es evaluado con respecto a un conjunto de *matches* en los *source models*. El segundo está compuesto de un conjunto de elementos donde cada uno especifica un *target type* desde el metamodelo y un conjunto de *bindings*. Un *binding* se refiere a una característica, i.e., un atributo, referencia o asociación. Las reglas especifican (1) cuáles *target elements* deben ser generados para cada *source element*, y (2) como los elementos generados son inicializados desde los *source elements* que coinciden, i.e., para cada elemento asociado a un *Variability*

Element, se verifican los contenidos del contexto con respecto a la lógica de las reglas de adaptación. La Ilustración 4.5 muestra la regla TaskUse, que utiliza el *helper* OptionalRule, el que a su vez utiliza otro *helper* llamado SystemArchitectureDefinition. Este resuelve mantener la tarea System architecture definition sólo si Project type posee el valor New development, o Interaction with other systems tiene el valor Complex.

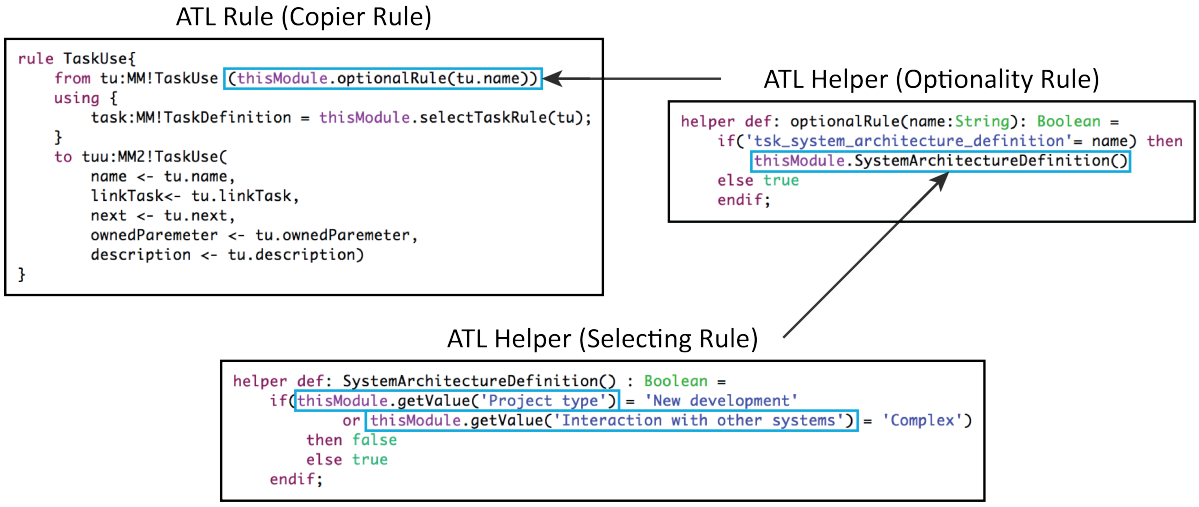
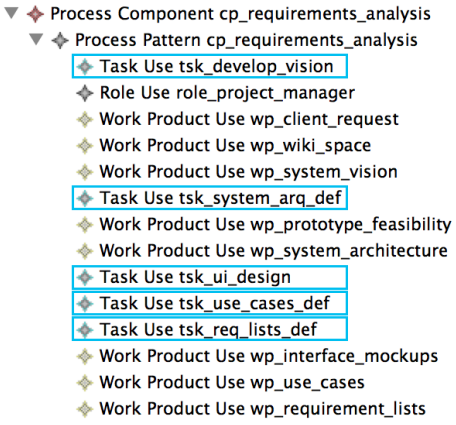
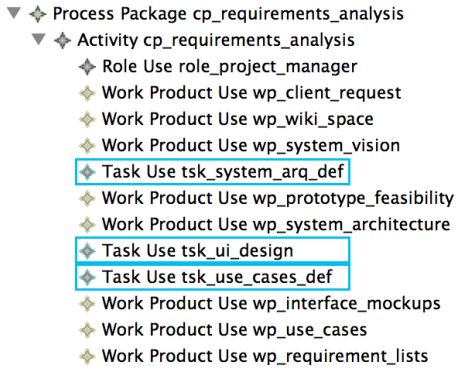


Ilustración 4.5: Ejemplo de reglas en ATL para transformación de modelos.

Actualmente, la transformación sólo considera a las tareas y actividades. Por lo tanto, el proceso generado posee sólo las tareas necesarias y ninguna extra, i.e., asegura una productividad y calidad óptimas para un proyecto con respecto a estos elementos. La Ilustración 4.6 destaca las tareas para dos extractos de ejemplo: (a) un proceso organizacional y (b) un proceso adaptado, para la actividad Análisis de Requerimientos de Mobius. En el segundo caso, no aparecen las tareas *Develop Vision* y *Requirements list definition*.



(a) Proceso organizacional.



(b) Proceso adaptado.

Ilustración 4.6: Extractos de procesos organizacional y adaptado para la actividad Análisis de Requerimientos de la empresa Mobius.

4.2. Procesos Predefinidos

Conocido el funcionamiento y los artefactos requeridos por la adaptación automática propuesta por ADAPTE, es posible diseñar una alternativa para evaluar su beneficio. Se propone un conjunto de procesos predefinidos, que inspirado en la estrategia *template based tailoring*, propone la selección de un proceso desde una familia de procesos, los cuales representan los tipos de proyectos más comunes de una empresa. Se ha mencionado que para responder el problema de esta tesis, se considerarán empresas que posean sus procesos organizacionales definidos para facilitar la generación de la familia de procesos.

En primer lugar, para deducir el contexto organizacional de una empresa se efectúa ingeniería reversa sobre su proceso organizacional. Esto se logra al inspeccionar sus anotaciones e identificar los *Context Attributes* y *Context Attributes Values*. Presente este artefacto, para determinar los tipos de proyectos más frecuentes realizados fueron consideradas las experiencias e impresiones de los investigadores que efectivamente participaron de la formalización, al interactuar directamente con los ingenieros de proceso de la empresa.

Luego se obtienen los procesos para los contextos definidos como más frecuentes utilizando la adaptación automática, i.e., obteniendo los procesos óptimos. A medida de que las empresas utilizan una familia de procesos de manera constante, también los mejoran hasta el punto en que resultan óptimos para el tipo de proyecto que abordan; incluso si no existe una definición formal. De esta forma, se establece un conjunto de pares contexto-proceso para la empresa, pero aún resta especificar cómo se realiza la selección de un proceso desde la familia de procesos.

Se ha establecido que evaluar el beneficio de la adaptación automática con respecto a los procesos predefinidos, comprende un conjunto de pasos secuenciales:

1. Definir un conjunto de contextos.
2. Obtener los procesos mediante una adaptación automática y una selección de uno de los procesos predefinidos.
3. Comparar correspondientemente los resultados.

Es importante que al considerar un conjunto acotado de tipos de proyectos, existe una alta probabilidad de que la plantilla seleccionada desde los procesos predefinidos posea tareas extras y tareas faltantes, con respecto al proceso óptimo de un contexto de entrada. La comparación entre procesos resultantes se realizará mediante la operación de *match* entre modelos. A continuación se detalla la selección de los procesos predefinidos y la comparación de los resultados entre las estrategias de adaptación.

Seleccionar el proceso más adecuado para un nuevo proyecto requiere que esté definido el contexto para el nuevo proyecto para compararlo con los contextos de los procesos predefinidos. Se seleccionará el proceso correspondiente al contexto con la mayor similitud. La Ilustración 4.7 muestra el diagrama correspondiente a los procesos predefinidos, donde se

utilizan dos operaciones para efectuar una adaptación: cálculo de similitud y obtención del proceso.

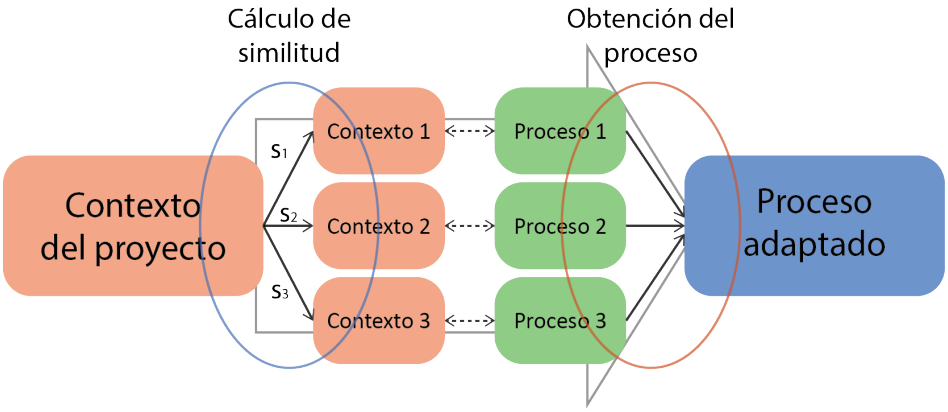


Ilustración 4.7: Procesos predefinidos, basada en la estrategia *template based tailoring*.

4.2.1. Cálculo de similitud

Se define similitud como el valor numérico entre cero y uno, obtenido al comparar de forma correspondiente los *Context Attributes Values* de dos contextos. Este valor es cero si todos los *Context Attributes* poseen diferentes *Context Attribute Values*, mientras que es uno si todos son iguales, y un valor intermedio si algunos valores difieren. El Algoritmo 1 indica que para obtener el valor de similitud se evalúan todos los *Context Attribute*, se suma uno a un contador cada vez que los *Context Attribute Values* sean iguales para un *Context Attribute*, y finalmente se divide la suma por la cantidad de *Context Attributes* evaluados. Se asume que los contextos provienen de un mismo contexto organizacional, i.e., la cantidad de *Context Attributes* es la misma para ambos.

Algorithm 1 Cálculo de similitud

```
1:  procedure SIMILITUD(c1, c2)                    ▷ c1 y c2 son contextos
2:      sim ← 0
3:      elements ← quantity of Context Attributes
4:      while attributes unchecked do
5:          if attribute value in c1 == attribute value in c2 then
6:              sim ← sim +1                       ▷ Suma 1 si los valores son iguales
7:          end if
8:          attribute ← next attribute
9:      end while
10:     return sim / elements                      ▷ Similitud entre dos contextos
11: end procedure
```

Para ilustrar el cálculo de similitud, es necesario recurrir al Cuadro 4.1 que muestra el contexto organizacional de Mobius, producto de la ingeniería reversa sobre su proceso organizacional: las *Dimensions*, *Context Attributes* y *Context Attributes Values* correspondientes.

Dimension	*Context Attribute*	*Context Attribute Values*
Proyecto	Tipo de proyecto	Nuevo desarrollo, Correctivo, No correctivo
Sistema	Tipo de sistema	Con garantía, Sin garantía
	Interacción con otros sistemas	Compleja, Simple
Equipo	Experiencia con la arquitectura	No, Sí
Producto	Usabilidad requerida	Alta, Baja
	Tipo de requerimiento	Reporte, Sin reporte
	Complejidad de la funcionalidad	Alta, Media, Baja
	Capa de acceso a datos	Sí, No
	Lógica del negocio	Sí, No
	Código compilable	Sí, No

Tabla 4.1: Mobius: Contexto organizacional.

Para ilustrar los pasos siguientes se considerarán los contextos o *Context Configurations* de ejemplo A, B y C, pertenecientes a la empresa Mobius. Se considerará que los contextos A y B corresponden a los procesos predefinidos, mientras que C será el contexto de un nuevo proyecto. En este ejemplo se tendrá en cuenta sólo la *Dimension* Sistema. La Tabla 4.2 resalta esta *Dimension*.

Dimension	*Context Attribute*	*Configuration* **A**	*Configuration* **B**	*Configuration* **C**
Proyecto	Tipo de proyecto	Nuevo desarrollo	Correctivo	No correctivo
Sistema	Tipo de sistema	Con garantía	Sin garantía	Con garantía
	Interacción con otros sistemas	Compleja	Simple	Simple
Equipo	Experiencia con la arquitectura	No	Sí	Sí
Producto	Usabilidad requerida	Alta	Baja	Alta
	Tipo de requerimiento	Sin reporte	Reporte	Sin reporte
	Complejidad de la funcionalidad	Alta	Media	Baja
	Capa de acceso a datos	Sí	No	No
	Lógica del negocio	No	Sí	No
	Código compilable	No	No	Sí

Tabla 4.2: Mobius: Contextos de ejemplos A, B y C.

La Tabla 4.3 muestra el resultado de aplicar el Algoritmo 1—considerando aisladamente la *Dimension* Sistema—a los pares de contextos (A, C), y (B, C); donde la similitud resultante en ambos casos es de 0,5. Sin perder generalidad ésto puede repetirse al considerar todos los *Context Attributes*; lo que indicaría que existen dos procesos predefinidos adecuados para un proyecto, aunque exista la intuición de que un proceso es más adecuado que otro. La situación anterior es producto de asumir que todos los *Context Attributes* son igualmente relevantes para la definición de un proyecto, pero la experiencia de otros investigadores y la propia al momento de formalizar un proceso indica lo contrario. La intuición del impacto en el proceso será representada mediante el peso de un *Context Attribute*.

Se define el peso de un *Context Attribute*, como un valor entre cero y uno calculado como la frecuencia relativa de aparición del elemento en todas las reglas de variabilidad presentes en el proceso organizacional. La suma de todos son uno. Esta definición captura el impacto de cada *Context Attribute*, mientras que refleja la experiencia de la SSE al momento de establecer las variabilidades. En la Tabla 4.4 se muestran los *Context Attributes* y los pesos calculados de esta forma.

	Similitud(A, C)	**Similitud(B, C)**
Tipo de sistema	0	1
Interacción con otros sistemas	1	0
Similitud	0.5	0.5

Tabla 4.3: Mobius: Cálculo de similitud sólo en la *Dimension* Sistema, utilizando el Algoritmo 1.

Dimension	***Context Attribute***	**Peso**
Proyecto	Tipo de proyecto	0.3
Sistema	Tipo de sistema	0.1
	Interacción con otros sistemas	0.1
Equipo	Experiencia con la arquitectura	0.1
Producto	Usabilidad requerida	0.1
	Tipo de requerimiento	0.1
	Complejidad de la funcionalidad	0.05
	Capa de acceso a datos	0.05
	Lógica del negocio	0.05
	Código compilable	0.05
	Total	1.0

Tabla 4.4: Mobius: Cálculo de pesos según la aparición de los *Context Attributes*.

Sin embargo, es posible refinar una vez más el cálculo del peso. Al inspeccionar el proceso organizacional es posible ver que si bien los *Context Attributes* aparecen repetidas veces en las anotaciones, lo importante es la cantidad de tareas a las que impactan, i.e., un *Context Attribute* es más relevante si impacta a una actividad compuesta de múltiples tareas, en vez de una sola tarea. Por esta razón, que el peso se calcula como la frecuencia relativa entre la cantidad de tareas que afecta un *Context Attribute*, con respecto a la cantidad total de tareas afectadas. La Tabla 4.5 muestra estos valores para Mobius, junto al cálculo de similitud para los pares de contextos (A, C), y (B, C).

Dimension	***Context Attribute***	**Peso**	**Similitud(A, C)**	**Similitud(B, C)**
Proyecto	Tipo de proyecto	0.394	0	0
Sistema	Tipo de sistema	0.098	0.098	0
	Interacción con otros sistemas	0.049	0	0.049
Equipo	Experiencia con la arquitectura	0.098	0	0.098
Producto	Usabilidad requerida	0.049	0.049	0
	Tipo de requerimiento	0.098	0.098	0
	Complejidad de la funcionalidad	0.033	0	0
	Capa de acceso a datos	0.066	0	0.066
	Lógica del negocio	0.066	0.066	0
	Código compilable	0.049	0	0
	Total	1.0	0.344	0.213

Tabla 4.5: Mobius: Cálculo de pesos según el impacto de los *Context Attributes* y cálculo de similitud entre las configuraciones de contextos (A, C) y (B, C).

El cuadro anterior muestra que la similitud (A, C) es mayor a la del par (B, C). De tener que seleccionar un proceso para el proyecto representado por el contexto C, el cálculo de

similitud indica que debe ser el proceso predefinido correspondiente al contexto A. De esta forma, el Algoritmo 2 muestra la modificación al Algoritmo 1, que considera los pesos de cada *Context Attribute*.

Algorithm 2 Cálculo de similitud con pesos

```
1: procedure SIMILITUD(c_1, c_2, w)          ▷ w Representa los pesos de los atributos
2:     sim ← 0
3:     while attributes unchecked do
4:         if attribute value in c_1 == attribute value in c_2 then
5:             sim ← sim + attribute weight                ▷ Suma el peso del atributo
6:         end if
7:         attribute ← next attribute
8:     end while
9:     return sim                                ▷ Similitud entre dos contextos
10: end procedure
```

Si bien esta modificación al cálculo de similitud considera la relevancia de los *Context Attributes*, aún es posible obtener similitudes iguales. Existen dos opciones para esta situación, (1) que un ingeniero de procesos elija uno de los procesos según el criterio que crea más conveniente, (2) que se realice otro procedimiento aparte de calcular la similitud. Para lo último se conjetura que un *merge* entre los procesos sería más adecuado que alguno de los procesos predefinidos por sí solo.

4.2.2. Obtención del proceso

Nuevamente, son tres son los resultados al calcular la similitud entre dos contextos considerando los pesos de sus *Context Attributes*, como indica el Algoritmo 2:

- Cero, donde se presume que el proceso predefinido es poco representativo.
- Uno, donde el proceso predefinido es efectivamente idéntico al proceso óptimo.
- Un valor intermedio, donde se infiere que los procesos predefinidos son similares al proceso óptimo para el contexto de entrada, pero no se sabe con exactitud en cuál grado. Existen tres subcasos:
  - Entregar el proceso asociado al valor de similitud más alto.
  - Dos procesos con similitudes iguales, donde se elige uno de forma arbitraria.
  - Dos procesos con similitudes iguales, donde se entrega un *merge* entre dichos procesos lo cual tiene la posibilidad de entregar un artefacto más cercano al proceso óptimo.

Como se ha indicado en la Sección 2.5, un *merge* entre modelos es una operación que consiste en construir un nuevo modelo que comprenda la unión de todos los elementos. Los procesos al ser instancias de procesos organizacionales, que en este caso conforman con el metamodelo de procesos eSPEM, permiten aplicar un *merge* entre procesos al considerarlos como conjuntos compuestos de *Process Elements*.

Ahora bien, una vez calculadas las similitudes entre el contexto de entrada y los contextos correspondientes a los procesos predefinidos, es necesario establecer un criterio que indique cuándo se realiza el *merge*. Se define umbral como un valor numérico entre cero y uno. Si el umbral es mayor al valor absoluto de la diferencia de las dos mayores similitudes, entonces se realiza el *merge* entre los procesos predefinidos correspondientes, en caso contrario no se realiza la operación y se utiliza el proceso de mayor similitud. La Ecuación 4.1 muestra este criterio.

$$\mid similitud_x - similitud_y \mid \leq umbral \tag{4.1}$$

Nuevamente se considera el ejemplo para la actividad Análisis de Requerimientos de Mobius y los contextos A, B y C; donde A y B serán los contextos correspondientes a los procesos predefinidos y C es el contexto de entrada. La similitud entre el par de contextos (A, C) posee un valor de 0.35, mientras que para (B, C) es de 0.25. Si el umbral tuviese un valor de 0.1, entonces como lo indica el criterio anterior, el proceso resultante para el contexto C será el *merge* entre los procesos A y B; si el umbral es menor a 0.1, se selecciona el proceso A como resultado.

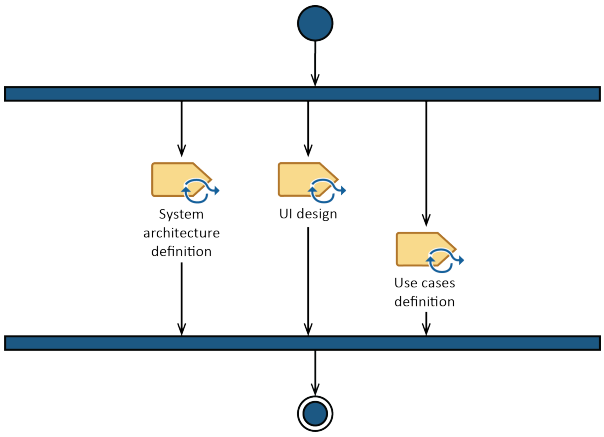
Para mostrar el efecto del umbral, la Ilustración 4.8 muestra los procesos automáticamente adaptados correspondientes a los contextos A, B y C, además del *merge* entre A y B. Al comparar el proceso C (c) con respecto al proceso A (a), proceso B (b) y el *merge* entre A y B (d), se puede decir que con respecto a las tareas extras y faltantes:

- El proceso A (a) posee una tarea extra (*System architecture definition*) y una faltante (*Develop vision*).
- El proceso B (b) posee una tarea extra (*Requirements list definition*) y tres faltantes (*Develop vision*, *UI design*, *Use cases definition*).
- El *merge* entre A y B (d) posee dos tareas extras (*System architecture definition*, *Requirements list definition*) y una faltante (*Develop Vision*).

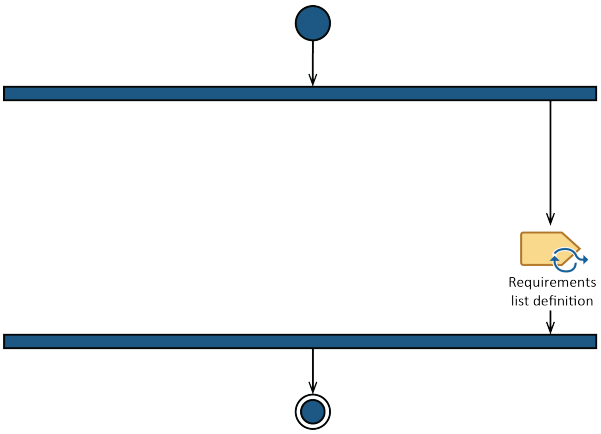
A primera vista y considerando sólo la actividad Análisis de Requerimientos, en ninguno de los casos se ha obtenido un proceso óptimo. La intuición de este mecanismo que incluye un *merge* entre los procesos es que al considerar el proceso completo se obtendrá un mejor resultado. Preliminarmente se piensa que la poca cantidad de procesos predefinidos o la magnitud del umbral, pueden afectar los resultados.

4.3. Comparación entre estrategias

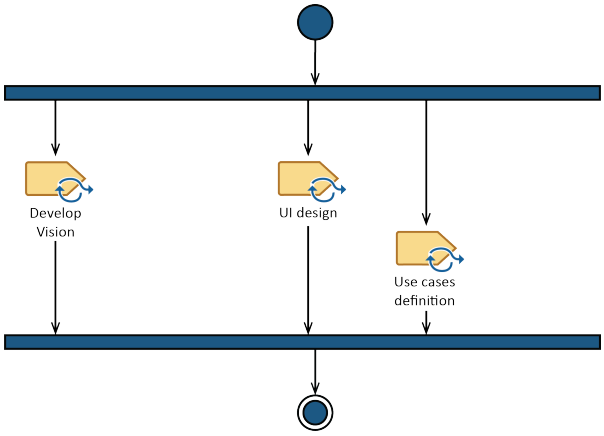
Como fue explicado en la Sección 3.1, beneficio de implementar una u otra estrategia de adaptación fue definido mediante en las métricas de tareas extras y tareas faltantes, al comparar los procesos obtenidos con las estrategias de adaptación para un mismo conjunto



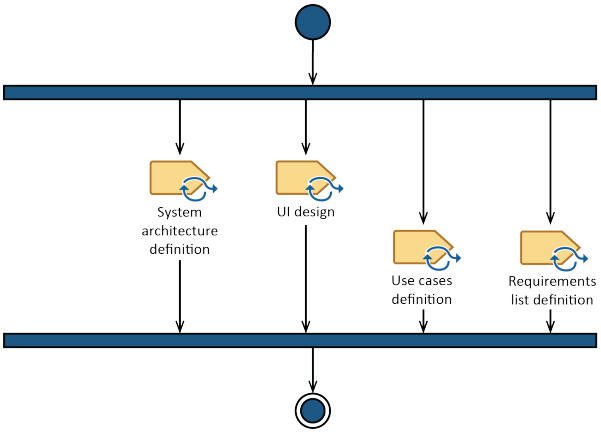
(a) Proceso adaptado A.



(b) Proceso adaptado B.



(c) Proceso adaptado C.



(d) Merge entre A y B.

Ilustración 4.8: Mobius: Actividad Análisis de Requerimientos para diferentes procesos.

de contextos. Lo anterior requiere realizar un *match* entre los procesos, i.e., compararlos con respecto a sus *Method Content Elements* e identificando cuántos y cuáles elementos coinciden.

Considerando C 4.8 (c) como un proceso óptimo, y el *merge* entre A y B 4.8 (d) como una plantilla de los procesos predefinidos. El Algoritmo 3 muestra los pasos para obtener las tareas extras y tareas faltantes de acuerdo con las métricas definidas en la Sección 3.1. Para la primera, se debe calcular la diferencia entre el *match* y las tareas correspondientes a la plantilla. En el ejemplo existen dos: *System architecture definition* y *Requirements list definition*. Mientras que las tareas faltantes son producto de la diferencia entre el *match* y el proceso óptimo. En el ejemplo sólo existe una: *Develop vision*.

Si bien el algoritmo 3 ilustra la comparación para una actividad, esto es aplicable para el proceso completo.

Los procesos poseen un conjunto común de tareas, entonces es posible interpretar a los elementos variables que no son necesarios como ineficiencias. Una tarea extra no es necesaria

Algorithm 3 Tareas extras y tareas faltantes para una actividad

```
1: procedure ExtrasYFaltantes(p1, p2)        ▷ p1 y p2 son procesos de software
2:     extra ←<>
3:     missing ←<>
4:     symetricDifference ← p1 Δ p2                          ▷ (p1 ∪ p2) \ (p1 ∩ p2)
5:     for element in p1 do                  ▷ Se evalúan todas las tareas en p1
6:         extra ← element ∩ symetricDifference
7:     end for
8:     for element in p2 do                  ▷ Se evalúan todas las tareas en p2
9:         missing ← element ∩ symetricDifference
10:    end for
11:    return | extra | + | missing |        ▷ Retorna las tareas extras y faltantes
12: end procedure
```

en el proceso y esta presente, mientras que una tarea faltante si es necesaria, pero no se encuentra en el proceso. La necesidad y presencia de la tarea dentro del proceso es lo que distingue a una tarea extra y una tarea faltante.

Ejecutar una tarea extra significa destinar tiempo y recursos en trabajo que no es necesario para el desarrollo del proyecto, lo que presumiblemente genera una productividad menor en el proyecto. Si bien no es ideal, ejecutar tareas extras es tolerable e incluso esperable dentro de los plazos y costos de un proyecto. Por otra parte, de existir una tarea faltante se ignoran pasos importantes en la realización de un proyecto y por ende, comprometer la calidad del un producto final. Ahora bien, depende de los estándares de calidad de la empresa las tareas faltantes son tolerables o no, aunque al ser uno de los factores más importantes en un producto de una SSE [3], indudablemente la holgura en este tipo de tareas es mucho menor que comparado a las tareas extras.

Como es posible cuantificar el beneficio, ahora es necesario responder a "cómo se evalúa". Dado que se han obtenido los contextos que representan los procesos típicos en una empresa y que es posible obtener el proceso óptimo mediante una adaptación automática, dichos procesos se considerarán como los procesos predefinidos. Asimismo, se ha descrito una manera en función de la mayor similitud para seleccionar un proceso, un mecanismo que posiblemente entregue un mejor resultado en caso de que existan dos similitudes cercanas y finalmente una manera comparar a nivel de *Method Content Elements*.

Para la mayoría de industria chilena, el tiempo utilizado para cerrar la venta de un proyecto es de tres meses, aunque también existen proyectos con ventas de hasta 9 meses. Asimismo, la liberación de dichos proyectos fluctúa entre 2 a 6 meses [3]. Por lo tanto, se comparará la estrategia de adaptación automática y los procesos predefinidos considerando un conjunto de cinco contextos inesperados y evaluando los resultados correspondientes entre cada estrategia, para cada empresa. Este horizonte de evaluación corresponde a la cantidad aproximada de proyectos completados durante un año por una SSE.

Por otro lado, las empresas Rhiscom y Mobius, poseen tres tipos de proyectos reflejados en los contextos organizacionales respectivos. El tipo de proyecto resulta ser el factor más importante al adaptar un proceso en una empresa. Según la investigación realizada por Hanssen [34], el tipo de proyecto determina el resto de las características en un proyecto como lo son el riesgo, el tipo de tecnología a utilizar, el conocimiento sobre el problema y el tamaño del proyecto. Por esta razón los contextos inesperados fueron elegidos para reflejar los tipos de proyectos en escenarios posibles de la empresa.

Capítulo 5

Validación

En el capítulo 4: "Diseño de la solución" se han descrito los pasos para obtener los procesos de software para la adaptación automática y los procesos predefinidos, y la forma de comparar los procesos con respecto sus tareas. Este capítulo presenta a las dos SSE chilenas consideradas para validar la solución: Rhiscom y Mobius, junto a la experimentación realizada, sus resultados y análisis.

5.1. Empresas

Usualmente las empresas de software utilizan especificaciones formales de procesos de software para especificar los comportamientos de las ejecuciones de sus procesos de software. Con la asistencia del equipo de ADAPTE, Rhiscom y Mobius han definido su contexto organizacional basado en el metamodelo SPCM, como su proceso organizacional de desarrollo software basado en el metamodelo eSPEM—utilizando la herramienta EPF Composer para indicar las variabilidades asociadas a sus tareas ya sean mandatorias, opcionales o alternativas.

Rhiscom[1] es una SSE de 17 años de antigüedad con un equipo de conformado por 46 personas, de los cuáles 40 conforman el equipo de desarrollo. Poseen más de 20 clientes a los que ofrecen la realización de proyectos, productos y servicios, especializándose en integración e implementación de hardware y software para los POS (*Point Of Sale*), utilizando tecnologías flexibles y específicas para cada proyecto. El nicho de Rhiscom comprende las áreas de:

- Software y hardware para POS.
- Sistema de automatización de testing en POS.
- Middleware transaccional.
- Sistema de gestión de procesos de negocio.
- Sistema de transacción digital de múltiples canales.

[1] http://www.rhiscom.com/ - Última visita: 18 de Diciembre 2014

- Sistema de monitoreo de operaciones en industria retail.

La estructura del ciclo de vida del proceso de software de Rhiscom está basada en las cuatro fases iterativas de RUP: Iniciación, Elaboración, Construcción y Transición. Sin embargo, a pesar de que esté basado en este enfoque, las tareas, roles y productos de trabajo parecen no mantener relación estricta con los correspondientes elementos de RUP. Este proceso ha sido diseñado por el *CEO* y fundador de la empresa. El proceso propuesto por RUP considera la necesidad de administrar los cambios y la adaptabilidad del proceso durante el desarrollo.

Mobius[2] es una SSE derivada de la empresa chilena DTS. Tiene tres años de experiencia y un equipo de 25 personas, donde sólo 11 de ellas ejecutan tareas de desarrollo de software. Ha participado del proyecto ADAPTE y actualmente es parte del proyecto GEMS[3]. Su nicho actual de mercado es el desarrollo de software y hardware integrado para el control y soporte del transporte público en Santiago, Chile, con énfasis en:

- Sistema de Control de Flotas para el transporte público.
- Sistema de Control de Acceso de pasajeros.
- Sistema de Videovigilancia para Buses.
- Sistema de Monitoreo de vehículos (GPS).
- Venta y Soporte Técnico de Ruteros Electrónicos para Buses.
- Venta y Soporte Técnicos de Torniquetes.

Mobius ha basado su proceso de software en OpenUP que es un "lean Unified Process", i.e., aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado al combinar prácticas de RUP con metodologías ágiles. El diseño del proceso fue realizado por el *Development Manager* de la empresa, que tiene amplia experiencia y ha liderado proyectos de software utilizando prácticas ágiles. La Ilustración 5.1 muestra este ciclo de vida basado en cuatro fases: Inicio, Elaboración, Construcción y Transición, cada una compuesta por un conjunto de *iterations* y *milestones*. Lo anterior provee a los *stakeholders* y al equipo de trabajo de visibilidad y manejo sobre las decisiones durante el proyecto.

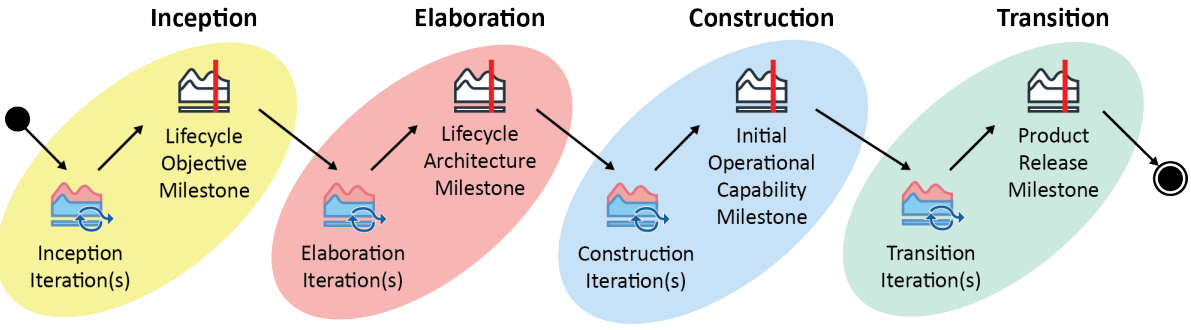


Ilustración 5.1: Diagrama de las fases de OpenUP.

[2]http://www.mobius.cl/ - Última visita: 18 de Diciembre 2014
[3]http://www.dcc.uchile.cl/gems/ - Última visita: 18 de Diciembre 2014

5.1.1. Descripción del proceso organizacional

Definida la comparación entre procesos es posible presentar una descripción sobre el proceso organizacional de cada empresa con respecto a sus tareas. Una tarea puede tener uno de los siguientes atributos, según su variabilidad: Mandatoria, Opcional o Alternativa. Las tareas opcionales se representan al tener una anotación asociada, que contiene la regla de decisión; mientras que las tareas alternativas tienen la regla asociada a un nodo de decisión. Es necesario mencionar que la notación utilizada en las figuras posteriores no es SPEM, sino una forma que ADAPTE ha utilizado para representar gráficamente los procesos, aunque EPF Composer permite definir formalmente las variabilidades.

La Ilustración 5.2 muestra el diagrama para la actividad Análisis de Requerimientos de la SSE Rhiscom, donde es posible apreciar tareas mandatorias (*Hold first meeting*, *Analize requirements*, *Verify requirements*, *Validate requirements*, *Approve requirements*) y alternativas (*Specify requirements in plain text*, *Specify requirements in use cases*, *Establish requirements baseline and test cases*, *Establish requirements baseline without test cases*). Este diagrama no posee tareas opcionales. Dado que los diagramas de actividades no poseen una notación específica para los elementos variables, y para hacer esto más claro en las empresas, el equipo de ADAPTE ha decidido utilizar UML; para expresar las tareas opcionales se utiliza una anotación y para las tareas alternativas se utiliza una anotación asociada a la disyunción.

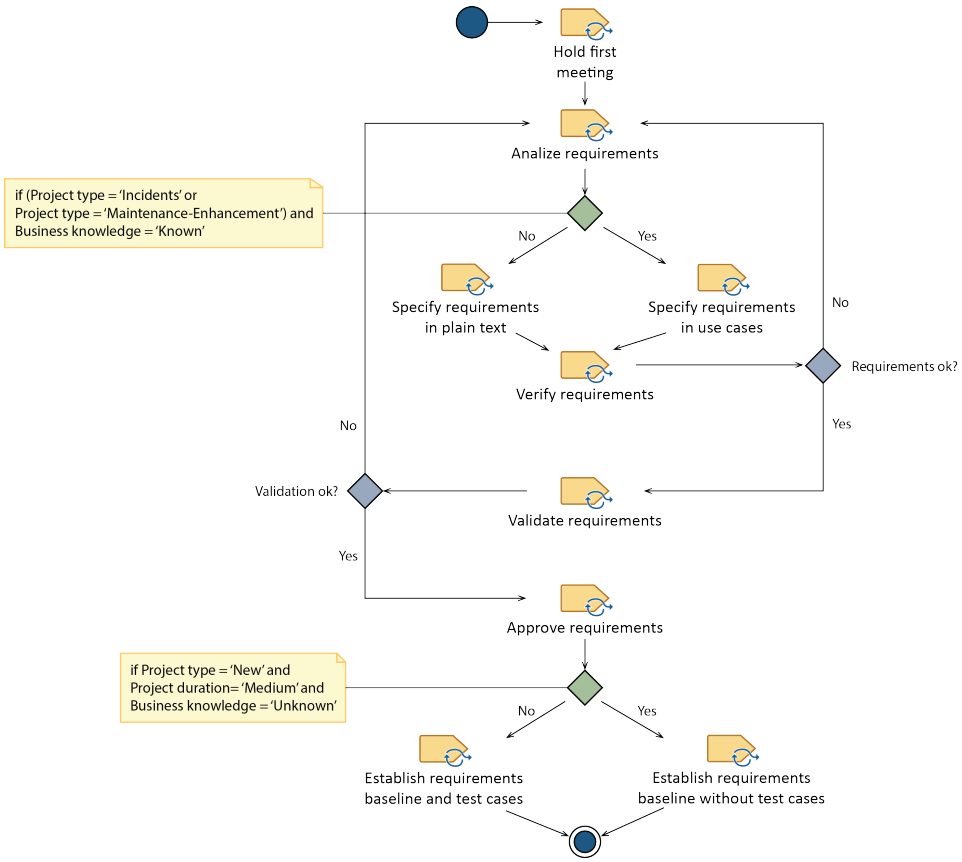


Ilustración 5.2: Rhiscom: Análisis de Requerimientos.

De igual manera, la Ilustración 5.3 muestra el diagrama de actividades para la actividad Análisis de Requerimientos de la empresa Mobius con sus tareas opcionales (*Develop vision*, *System architecture definition*, *UI design*) y alternativas (*Use cases definition*, *Requirements list definition*). Este diagrama no posee tareas mandatorias.

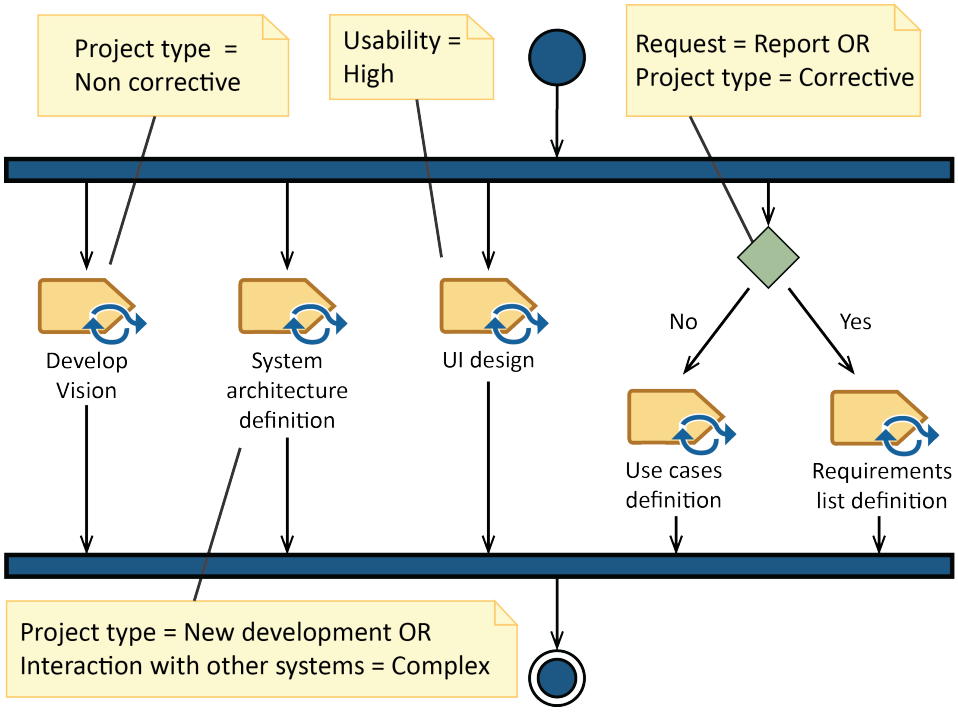


Ilustración 5.3: Mobius: Análisis de Requerimientos.

A continuación se muestra el detalle de los procesos organizacionales de Rhiscom y Mobius, que poseen una estructura similar al seguir una versión propia y simplificada de RUP, y OpenUP respectivamente. La Tabla 5.1 muestra el detalle de Rhiscom en sus cuatro fases y además de manera consolidada; cuenta con un total de 35 tareas en su proceso: 28 mandatorias, 5 opcionales y sólo 2 alternativas.

Fases proceso organizacional Rhiscom								**Tareas del proceso**	
Iniciación		Elaboración		Construcción		Transición			
Mandatorias	4	Mandatorias	5	Mandatorias	11	Mandatorias	3	Mandatorias	23
Opcionales	2	Opcionales	6	Opcionales	1	Opcionales	0	Opcionales	9
Alternativas	0	Alternativas	4	Alternativas	0	Alternativas	0	Alternativas	4
Total fase	6	Total fase	15	Total fase	12	Total fase	3	**Total**	**36**

Tabla 5.1: Rhiscom: Descripción del proceso organizacional.

Por otro lado, la Tabla 5.2 muestra que el proceso organizacional de Mobius posee una granularidad más fina comparada con Rhiscom, con 106 tareas divididas en 64 mandatorias, 30 opcionales y 12 alternativas.

Fases proceso organizacional Mobius								**Tareas del proceso**	
Iniciación		Elaboración		Construcción		Transición			
Mandatorias	8	Mandatorias	0	Mandatorias	27	Mandatorias	29	Mandatorias	64
Opcionales	8	Opcionales	0	Opcionales	11	Opcionales	11	Opcionales	30
Alternativas	2	Alternativas	6	Alternativas	2	Alternativas	2	Alternativas	12
Total fase	18	Total fase	6	Total fase	40	Total fase	42	**Total**	**106**

Tabla 5.2: Mobius: Descripción del proceso organizacional.

Con respecto a los elementos variables en ambos procesos organizacionales, en Rhiscom 13 de 36 tareas son variables (36,11 %) mientras que para Mobius 42 de 106 tareas son variables (39,62 %). Lo anterior no permite realizar inferencia alguna con respecto a la cantidad de tareas extras y tareas faltantes entre los procesos adaptados para Rhiscom o Mobius, ya que los valores de variabilidad son similares.

5.1.2. Descripción del contexto organizacional

En un principio, al no estar disponibles los contextos organizacionales para Rhiscom ni Mobius, se ha realizado ingeniería inversa sobre las variabilidades concretas, establecidas para dichos procesos al inspeccionar las anotaciones en sus diagramas de actividades. Para cada empresa, y basándose en SPCM, se han reconstruido los contextos organizacionales al identificar sus *Context Attributes* y los respectivos *Context Attribute Values*, dispuestos en un conjunto de *Dimensions*. La Tabla 5.3 muestra el resultado de este procedimiento para la empresa Rhiscom además de incluir los pesos de sus atributos, calculados como se indica de la Sección 4.2.1.

Dimension	***Context Attributes***	***Context Attributes Values***	**Peso**
Tipo	Tipo de proyecto	Nuevo desarrollo, Incidental, Mantención-Mejora, Mantención-Corrección, Mantención-Adaptación.	0.342
Duración	Duración del proyecto	Alta, Media, Baja.	0.263
Equipo	Tamaño del equipo	Grande, Medio, Bajo.	0.053
Negocio	Conocimiento del negocio	Conocido, Abordable, Desconocido.	0.342

Tabla 5.3: Rhiscom: Contexto organizacional y peso de sus *Context Attributes*.

En el caso de Mobius, por su nivel más detallado de granularidad y al tener una mayor cantidad de elementos variables en su proceso organizacional comparado al de Rhiscom, su contexto organizacional posee un mayor número de variables de contexto. La Tabla 5.4 describe el contexto organizacional resultante.

Dimension	*Context Attributes*	*Context Attributes Values*	**Peso**
Proyecto	Tipo de proyecto	Nuevo desarrollo, Correctivo, No correctivo.	0.394
Sistema	Tipo de sistema	Con garantía, Sin garantía.	0.098
	Interacción con otros sistemas	Compleja, Simple.	0.049
Equipo	Experiencia con la arquitectura	Sí, No.	0.098
Producto	Usabilidad requerida	Alta, Baja.	0.049
	Tipo de requerimiento	Reporte, Sin reporte.	0.098
	Complejidad de la funcionalidad	Alta, Media, Baja.	0.033
	Capa de acceso a datos	Sí, No.	0.066
	Lógica del negocio	Sí, No.	0.066
	Código compilable	Sí, No.	0.049

Tabla 5.4: Mobius: Contexto organizacional y peso de sus *Context Attributes*.

5.2. Experimentación

Ya mostrados el proceso y contexto organizacional para ambas empresas, es posible describir el experimento realizado que permite responder el problema planteado: *"¿cuál es el beneficio para una empresa de software de implementar una estrategia de adaptación automática de procesos?"* Entonces es necesario describir los pasos realizados en cada una de las empresas durante la experimentación: la obtención de los datos y la comparación entre ambas estrategias, para luego reportar los resultados obtenidos.

Ya que se desea evaluar tanto la productividad como la calidad de los procesos predefinidos con respecto a los obtenidos mediante la adaptación automática de procesos, se considerará que un proceso posee mayor productividad al tener una menor cantidad de tareas extras ya que no efectúa tareas innecesarias que pueden ser medidas por ejemplo, en horas hombre. Un proceso de mayor calidad refiere a una menor cantidad de tareas faltantes, ya que no se dejan de lado tareas fundamentales en la realización del proyecto.

Cabe recordar que para el alcance de esta tesis, la estrategia de adaptación automática obtiene un proceso adaptado al efectuar una transformación donde un proceso organizacional es ajustado al contexto de un proyecto; se asume por construcción este proceso adaptado es óptimo. Los procesos predefinidos definen un conjunto de procesos o plantillas, que representados por sus contextos, corresponden a los procesos más frecuentemente efectuados dentro de la empresa. El proceso adaptado es seleccionado según la mayor similitud, que es calculada comparando el contexto del nuevo proyecto y los correspondientes a las plantillas definidas. Con lo anterior es posible ilustrar la experimentación realizada en cada empresa mediante los siguientes pasos:

1. Ya definidos los contextos de plantillas (CP), se define un conjunto de contextos inesperados (CI) que podrían representar otros escenarios de proyectos posibles en la empresa.
2. Obtener los procesos adaptados para cada contexto inesperado utilizando ambas estrategias de adaptación, i.e., la adaptación automática (AA) y los procesos predefinidos (PP).
3. Comparar los procesos obtenidos en cada una de las estrategias, de forma correspondien-

te. Para cada contexto inesperado (i $\in CI$) se obtiene el proceso adaptado automáticamente (AA_{i}) y se selecciona la plantilla correspondiente (PP_{i}). Éstos son comparados para obtener la cantidad de tareas extras y tareas faltantes ($AA_{\text{i}} \triangle PP_{\text{i}}$).

El número de contextos inesperados (CI) fueron definidos y acotado a sólo 5. Si bien es posible generar todos los contextos posibles, efectuar las adaptaciones correspondientes y obtener todos los resultados, es más valioso evaluar contextos utilizados en escenarios de proyectos reales por las SSE.

5.2.1. Rhiscom

Las filas de la Tabla 5.5 muestran el listado de contextos para Rhiscom: tanto los contextos de plantillas, e.g., CP: contextos A, B y C, como los contextos inesperados, e.g., CI: contextos D, E, F, G y H. Basándose en la información de la Tabla 5.3, en las columnas se listan los *Context Attributes* y las celdas muestran los *Context Attributes Values* correspondientes a cada contexto.

		Context Attributes			
		Tipo de proyecto	Duración del proyecto	Tamaño del equipo	Conocimiento del negocio
Contextos de plantillas	Contexto A	Nuevo desarrollo	Grande	Mediano	Desconocido
	Contexto B	Mantención-corrección	Pequeño	Pequeño	Conocido
	Contexto C	Incidentes	Pequeño	Pequeño	Conocido
Contextos inesperados	Contexto D	Nuevo desarrollo	Mediano	Mediano	Abordable
	Contexto E	Mantención-mejora	Mediano	Mediano	Conocido
	Contexto F	Mantención-corrección	Pequeño	Pequeño	Abordable
	Contexto G	Mantención-adaptación	Pequeño	Pequeño	Abordable
	Contexto H	Incidentes	Pequeño	Mediano	Desconocido

Tabla 5.5: Rhiscom: Listado de contextos.

Por otro lado, la Tabla 5.6 muestra el cálculo de similitud para Rhiscom, donde las filas listan los contextos inesperados (CI), mientras que las columnas muestran a los contextos de plantillas (CP). Las celdas indican los resultados del cálculo de similitud según el algoritmo 2 descrito en la Sección 4.2.1, según pesos de cada *Context Attribute* indicados en la Tabla 5.3.

Las celdas con los valores de mayor similitud son resaltadas e indican que el proceso correspondiente a la plantilla que debe ser utilizada. Si existen dos valores iguales y mayores para una fila, son resaltadas ambas celdas. e.g., los contextos E y G, pero no existe manera *apriori* de decidir cuál de los procesos predefinidos se debe aplicar. Para el proyecto representado por el contexto inesperado D se utilizará el proceso correspondiente al contexto A, debido a su mayor similitud. Respectivamente, el contexto E puede utilizar los procesos B o C, F se utilizará B, G puede utilizar los procesos B o C, y para H se utilizará el proceso correspondiente al contexto C.

		Contextos de plantillas		
		Contexto A	Contexto B	Contexto C
Contextos inesperados	Contexto D	0.395	0.000	0.000
	Contexto E	0.053	0.342	0.342
	Contexto F	0.000	0.658	0.316
	Contexto G	0.000	0.316	0.316
	Contexto H	0.395	0.263	0.605

Tabla 5.6: Rhiscom: Cálculo de similitud.

La Tabla 5.7 muestra la comparación en términos de tareas extras y tareas faltantes entre los procesos predefinidos (PP), e.g., procesos A, B y C, y los procesos óptimos para los contextos inesperados (AA), e.g., procesos D, E, F, G y H, obtenidos desde la adaptación automática. Se resaltan los valores de tareas extras y tareas faltantes, correspondientes a los procesos seleccionados por la estrategia de procesos predefinidos.

Al comparar el proceso óptimo para el contexto inesperado D y la plantilla seleccionada A, se encuentran cero tareas extras y faltantes, i.e., los procesos son iguales y el seleccionado es el óptimo. Para el proceso óptimo del contexto inesperado E, el proceso seleccionado B carece de una tarea, mientras que el proceso C posee una tarea extra. Para el proceso óptimo de F, la plantilla seleccionada B es óptima. En el caso del proceso óptimo G, en la plantilla B falta una tarea y la plantilla C es óptima. Finalmente, el proceso óptimo H es igual a la plantilla C.

	Plantillas					
	Proceso A		Proceso B		Proceso C	
Procesos óptimos	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes
Proceso D	0	0	0	1	0	0
Proceso E	1	0	0	1	1	0
Proceso F	4	0	0	0	4	0
Proceso G	0	0	0	1	0	0
Proceso H	0	0	0	1	0	0

Tabla 5.7: Rhiscom: Cálculo de tareas extras y tareas faltantes.

Finalmente, la Tabla 5.8 muestra las tareas extras y tareas faltantes para los *merge* entre los procesos de la estrategia de procesos predefinidos, comparados con los procesos óptimos para los contextos inesperados para Rhiscom. Dada la Tabla 5.6 y considerando un umbral cero, es necesario obtener los *merges* entre los procesos (B, C) para los contextos E y G. Sin embargo, se presentan todos los casos para ilustrar el impacto del *merge*.

En el caso del proceso D los tres *merge* son óptimos, ya que individualmente las plantillas A y C lo son. Para el proceso E, la cantidad de tareas faltantes es mitigada con respecto al óptimo, pero las plantillas A y C propagan su error. Lamentablemente para el proceso F que ya cuenta con un óptimo, el *merge* lo único que provoca es un aumento en la cantidad

de tareas extras. Para los procesos G y H, se mitigan las tareas faltantes y los *merge* son óptimos.

	Plantillas					
	Merge AB		Merge AC		Merge BC	
Procesos óptimos	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes
Proceso D	0	0	0	0	0	0
Proceso E	1	0	1	0	1	0
Proceso F	4	0	4	0	4	0
Proceso G	0	0	0	0	0	0
Proceso H	0	0	0	0	0	0

Tabla 5.8: Rhiscom - Merge: Cálculo de tareas extras y tareas faltantes.

5.2.2. Mobius

Análogamente para Mobius, la Tabla 5.9 muestra en sus filas la lista de contextos para los procesos predefinidos, e.g., CP: contextos A, B y C, y para los contextos inesperados, e.g., CI: contextos D, E, F, G y H. Según lo indicado en la Tabla 5.4, las columnas listan los *Context Attributes*; las celdas muestran los *Context Attributes Values* correspondientes a cada contexto.

		Tipo de proyecto	Tipo de sistema	Interacción con otros sistemas	Experiencia en la arquitectura	Usabilidad
Contextos de plantillas	Contexto A	Nuevo desarrollo	Garantía	Compleja	No	Alto
	Contexto B	Correctivo	No garantía	Simple	Si	Bajo
	Contexto C	No correctivo	Garantía	Simple	Si	Alto
Contextos inesperados	Contexto D	Nuevo desarrollo	No garantía	Simple	No	Bajo
	Contexto E	Correctivo	Garantía	Compleja	Si	Bajo
	Contexto F	No correctivo	Garantía	Simple	Si	Alto
	Contexto G	No correctivo	No garantía	Compleja	Si	Alto
	Contexto H	Correctivo	Garantía	Compleja	No	Bajo

		Requerimiento	Complejidad de la funcionalidad	Capa de acceso a datos	Lógica del negocio	Código fuente
Contextos de plantillas	Contexto A	Sin reporte	Alto	Si	No	No
	Contexto B	Reporte	Medio	No	Si	No
	Contexto C	Sin reporte	Bajo	No	No	Si
Contextos inesperados	Contexto D	Reporte	Alto	Si	Si	Si
	Contexto E	Reporte	Medio	No	No	Si
	Contexto F	Reporte	Alto	No	No	No
	Contexto G	Reporte	Medio	Si	Si	No
	Contexto H	Sin reporte	Bajo	Si	Si	Si

Tabla 5.9: Mobius: Listado de contextos.

De forma similar a Rhiscom, las filas de la Tabla 5.10 muestran los contextos inesperados (CI) y las columnas los contextos de plantillas (CP). De acuerdo con la Tabla 5.4 donde se indican los pesos de cada *Context Attribute*, las celdas contienen el cálculo de similitud según

el algoritmo 2 descrito en la Sección 4.2.1. Las celdas de mayor similitud son resaltadas e indican que el proyecto representado por el contexto inesperado D utilizará el proceso correspondiente al contexto de plantillas A. Respectivamente, el contexto E utilizará el proceso B, F utilizará C, G utilizará C, y para H se utilizará el proceso correspondiente al contexto de plantillas B.

		Contextos de plantillas		
		Contexto A	Contexto B	Contexto C
Contextos inesperados	Contexto D	0.591	0.360	0.098
	Contexto E	0.213	0.738	0.377
	Contexto F	0.295	0.360	0.820
	Contexto G	0.213	0.442	0.541
	Contexto H	0.409	0.509	0.278

Tabla 5.10: Mobius: Cálculo de similitud.

En el caso de Mobius, la Tabla 5.11 muestra la comparación, en términos de tareas extras y tareas faltantes, entre los procesos correspondientes a los contextos de plantillas, e.g., procesos A, B y C, y los procesos óptimos para los contextos inesperados, e.g., procesos D, E, F, G y H. Se resaltan los valores de tareas extras y tareas faltantes, correspondientes a los procesos elegidos por la estrategia de procesos predefinidos.

Siguiendo lo indicado por la Tabla 5.10, el resultado de comparar el proceso óptimo para el contexto inesperado D y la plantilla seleccionada A son seis tareas extras y seis tareas faltantes; al comparar el óptimo E y el proceso seleccionado B se encuentran 10 extras y seis faltantes, para F y C son seis extras y cinco faltantes, para G y C son seis extras y once faltantes, y para el proceso H y el proceso B son seis extras y siete faltantes.

	Plantillas					
	Proceso A		Proceso B		Proceso C	
Procesos óptimos	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes
Proceso D	6	6	6	10	21	14
Proceso E	13	6	10	7	21	7
Proceso F	12	18	4	14	6	5
Proceso G	6	18	0	16	6	11
Proceso H	9	6	6	7	21	11

Tabla 5.11: Mobius: Cálculo de tareas extras y tareas faltantes.

Finalmente, la Tabla 5.12 muestra las tareas extras y tareas faltantes para los *merge* entre los correspondientes a los procesos predefinidos, comparados con los procesos óptimos para los contextos inesperados de Mobius. Dada la Tabla 5.10 y considerando un umbral cero, no es necesario obtener los *merges* para ningún contexto. Sin embargo, se presentan todos los casos para ilustrar el efecto del *merge* y si existe algún caso con una menor cantidad de tareas extras o faltantes.

En este caso, mayoritariamente la cantidad de tareas extras se ha incrementado en todos los casos y sólo ocasionalmente se ha mantenido constante, e.g., proceso F comparado al merge AB, proceso D, G y H comparados al merge BC. Por otro lado, la cantidad de tareas faltantes ha sido mitigada de forma generalizada.

	Plantillas					
	Merge AB		Merge AC		Merge BC	
Procesos óptimos	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes	Tareas extras	Tareas faltantes
Proceso D	12	3	21	3	21	7
Proceso E	19	3	28	3	25	4
Proceso F	12	9	15	3	10	2
Proceso G	6	9	9	3	6	4
Proceso H	15	3	24	3	21	4

Tabla 5.12: Mobius - Merge: Cálculo de tareas extras y tareas faltantes.

5.3. Análisis de resultados

La comparación realizada con información empírica comprueba un resultado teóricamente esperado: las soluciones obtenidas en función de los procesos predefinidos prácticamente siempre difieren del resultado de la adaptación automática, considerado como óptimo. Lo anterior no depende totalmente del hecho de considerar como óptimos los resultados de la adaptación automática, sino que también de la forma de definir y ejecutar los procedimientos de Cálculo de Similitud (4.2.1) y Obtención del Proceso (4.2.2).

Sin embargo, es posible notar en la Tabla 5.7, que las plantillas A y C obtienen resultados óptimos aunque sus contextos posean distintos *Context Attribute Values* para sus *Context Attributes*, como lo indica la Tabla 5.5. El contexto A posee los valores Tipo de proyecto-Nuevo Desarrollo, Duración del proyecto-Grande, Tamaño del equipo-Mediano y Conocimiento del negocio-Desconocido, mientras que el contexto C posee los valores Incidentes, Pequeño, Pequeño y Conocido, respectivamente. Esta situación ocurre debido a que las condiciones de variabilidad son compuestas de múltiples disyunciones que abarcan diferentes escenarios, e.g., a pesar de lo diferentes que sean dos contextos, se utiliza exactamente el mismo proceso. La posibilidad de que dos proyectos con distintos contextos puedan utilizar el mismo proceso adaptado automáticamente, es un hecho inusual que sugiere falencias en la definición del proceso organizacional y sus variabilidades. Esto no sucede en el caso de Mobius, donde todas las plantillas y todos los procesos adaptados automáticamente son distintos.

La Tabla 5.7 indica que para los contextos D, F y H de Rhiscom, la plantilla seleccionada según la mayor similitud corresponde al proceso óptimo, i.e., ausencia de tareas extras y tareas faltantes; para los contextos E y G no es posible seleccionar uno de los procesos B o C, ya que en ambos casos las similitudes son iguales, aunque ambas plantillas "candidatas" son buenas aproximaciones, lo que se cree es debido a como han sido definidos los elementos variables del

proceso. La Tabla 5.11 muestra para Mobius que la plantilla seleccionada correspondiente al contexto D, posee la menor cantidad de tareas extras y tareas faltantes entre las alternativas disponibles. Las plantillas seleccionadas para los contextos E, F y H, son efectivamente más ajustadas que las otras, aunque no se encuentran distribuidas de manera que las tareas extras y tareas faltantes sean menores. Finalmente, la plantilla en el caso de G es menos ajustada.

En los Cuadros 5.6 y 5.10 es posible apreciar cómo el cálculo de las similitudes articula el impacto de cada *Context Attribute* con respecto a la selección de los procesos predefinidos. Por otro lado, la Tabla 5.5 indica que en el caso de Rhiscom los *Context Attributes* Tipo de proyecto y Conocimiento del negocio poseen los pesos más grandes, i.e., aparecen en todas las variabilidades existentes. Por otro lado, al comparar los contextos inesperados con los contextos de las plantillas seleccionadas, al menos un *Context Attributes Value* de Tipo de proyecto o Conocimiento del negocio, es igual. La Tabla 5.9 muestra que de forma análoga para Mobius que Tipo de proyecto posee el mayor peso. En los procesos predefinidos lo anterior indica que la selección según la mayor similitud basada en el peso, está influenciada por el conjunto de *Context Attributes* que poseen un mayor peso, y que a su vez restan importancia a los otros *Context Attributes*. El caso en que todos los *Context Attributes* posean un mismo peso es posible, pero dadas las empresas consideradas no ha ocurrido algo de dicha naturaleza.

Se ha evaluado la productividad y calidad de los procesos predefinidos con respecto a la adaptación automática de procesos al comparar correspondientemente sus procesos adaptados. La Tabla 5.7 muestra que en los procesos D, F, G y H de Rhiscom se alcanza el proceso óptimo mediante los procesos predefinidos; la plantilla correspondiente al contexto E tendría una pérdida de productividad o calidad, en sólo una tarea. Por otro lado, la Tabla 5.11 muestra que para Mobius esta situación es diferente ya que ninguna de las plantillas provee un proceso óptimo, lo que se traduce en una pérdida de productividad, i.e., al destinar tiempo en efectuar tareas que no son necesarias para lograr un producto entregable, y calidad, i.e., al no efectuar tareas necesarias para obtener un producto satisfactorio.

Es necesario discutir el mecanismo de *merge* en los procesos predefinidos. En el caso de Rhiscom, la Tabla 5.6 muestra que es posible hacer *merge* en dos casos: para el contexto E y el contexto G, ambos con respecto a los procesos B y C que poseen similitudes iguales en ambos casos. Es posible apreciar en la Tabla 5.7, que para el proceso óptimo E, el proceso B tiene una tarea faltante y el proceso C tiene una tarea extra. Análogamente, para el proceso óptimo G, el proceso B posee una tarea faltante y el proceso C es óptimo. En estos casos no habría una manera decisiva para seleccionar un proceso sin el mecanismo de *merge*. Ahora bien, la Tabla 5.8 muestra los resultados de comparar los procesos óptimos con respecto a los procesos producidos por el *merge*. Para el proceso óptimo E, el *merge* entre B y C posee una tarea extra y ninguna faltante. En cambio, el proceso óptimo G y el *merge* entre B y C son iguales.

Para Mobius, si consideramos un umbral igual a cero, no es necesario efectuar un *merge* entre los procesos ya que no existen dos similitudes que sean iguales, pero al considerar umbrales suficientemente grandes es posible aplicar los procesos obtenidos del *merge*. Según la Tabla 5.10 y considerando un umbral con valor 0.1, es posible observar que al contexto G

le corresponde un *merge* entre los procesos B (similitud de 0.442) y C (similitud de 0.541). Análogamente, para el contexto H corresponde un *merge* entre los procesos A (similitud 0.409) y B (similitud de 0.509). Según la mayor similitud y los Cuadros 5.11 y 5.12, para el contexto G corresponde el proceso C con 6 tareas extras y 11 faltantes, y el *merge* entre B y C posee 6 tareas extras y 4 faltantes, i.e., disminuyen en 7 las tareas faltantes. Para el contexto H corresponde el proceso B con 6 tareas extras y 7 faltantes, y el *merge* entre A y B posee 15 tareas extras y 3 faltantes, i.e., aumentan en 9 las tareas extras y en 4 las tareas faltantes. Si bien los datos obtenidos muestran que las tareas faltantes se han visto mitigadas, no es el caso general para las tareas extras. Es posible obtener un proceso más ajustado a partir de un *merge*, pero lamentablemente es un caso infrecuente.

En el caso de Rhiscom se observa que los procesos predefinidos permiten obtener resultados óptimos para los contextos D, F y H, i.e., tres de los cinco casos considerados; mientras que para el contexto G es posible obtener el óptimo mediante el *merge*. El único contexto inesperado que no se alcanza el óptimo es el contexto E, aunque por sólo una tarea extra. Para todos los contextos inesperados de Mobius, no existe proceso predefinido que sea óptimo. Sin embargo, la similitud efectivamente obtiene el proceso con la menor cantidad absoluta de tareas, i.e., la suma de tareas extras y tareas faltantes. Ahora bien, el *merge* ha mostrado disminuir la cantidad de tareas faltantes en todos los casos para ambas empresas, con el costo de incluir una cantidad considerable de tareas extras; lo que finalmente afectaría la productividad del proyecto.

Los resultados obtenidos se deben a la naturaleza de los datos: la cantidad de empresas evaluadas, los procesos organizacionales y cómo han sido definidos. Por otro lado, el método utilizado para evaluar ambas estrategias de adaptación de procesos de software es una de las mayores contribuciones de este trabajo: la decisión de recrear la ejecución de los proyectos en las empresas, la selección de un proceso predefinido, además de la comparación entre procesos y uso de *merge* entre procesos para extender la solución inicial.

5.4. Amenazas a la validez

En todo estudio con datos empíricos, las amenazas a la validez deben ser identificadas y dirigidas de forma sistemática. En primer lugar, si bien existen SSE chilenas que utilizan procesos de software, no es clara la cantidad de ellas que utilizan estrategias de adaptación automática de procesos en sus proyectos. Dentro del alcance de esta tesis se ha asumido que el uso de plantillas es generalizado y por eso se ha considerado para comparar con respecto a la adaptación automática de procesos de ADAPTE.

Por otro lado, se ha asumido de antemano que la estrategia de adaptación automática obtiene mejores resultados que los que otorgaría la estrategia de procesos predefinidos. Sin embargo, al estar bajo el contexto del proyecto ADAPTE, las razones de esta presunción son variadas:

- Se realizó la definición de un procedimiento formal para definir y modelar la infraestructura y uso de la adaptación automática [36].
- Se han efectuado refinamientos y especificaciones sobre las técnicas para su uso [39, 40] y su factibilidad [38].
- Para mejorar su uso, se han generado diferentes herramientas que incrementen la usabilidad de la estrategia, ya sea para definir los contextos [63], las reglas de transformación [76] e incluso analizar el proceso organizacional [37].
- Se ha validado la estrategia, en más de una SSE [73].
- Se han considerado otras aristas, como la mejora en su adopción [9], incluso en un ambiente mucho más complejo [10].

Una fuente de amenazas a la validez proviene del nivel de involucramiento en la definición de los procesos organizacionales por parte del autor. Sin embargo, en esta tesis no se ha participado en la definición de los procesos organizacionales, ni con los funcionarios para definir o aplicar los procesos organizacionales de software. Ahora bien, se han recopilado datos empíricos para establecer los contextos de plantillas que describiesen mejor los escenarios de la empresa. Asimismo, la definición de tareas extras y tareas faltantes fueron validadas por los funcionarios de las empresas.

Las SSE fueron elegidas ya que trabajan en el ambiente chileno y cuentan con sus procesos formalizados. Ambas empresas se diferencian fuertemente en el nicho que trabajan y el tiempo que demoran en entregar sus proyectos, i.e., *time-to-market* (TTM). Estas diferencias agregan diversidad al comparar las estrategias de adaptación. En esta tesis no fue posible conseguir una muestra de datos suficientemente grande y diversa, que permita concluir con resultados estadísticamente válidos sobre la comparación entre procesos en SSE. Una mayor cantidad de empresas presentaría una mayor variedad en diferentes aspectos como: tamaño y antigüedad de la empresa, tamaño y formación de los equipos, tamaño y tipo de proyectos a realizar y madurez de los procesos utilizados, por nombrar algunos. Es posible representar todos los factores mencionados gracias a los metamodelos de procesos y contextos organizacionales; la limitación está en la existencia de SSE que posean su proceso formalizado, que utilicen una estrategia de líneas de procesos con variaciones o que tengan la intención de formalizar su proceso. En el caso de existir, para obtener la colaboración de estas empresas, i.e., ceder su información o aplicar nuevas estrategias, se requeriría un esfuerzo considerable.

En todo proyecto es posible que cambie el escenario inicial debido a factores externos, e.g., disponibilidad del personal, recortes en el presupuesto, salida o cambios en los clientes. La adaptación automática puede redefinir el contexto y obtener un nuevo proceso muy fácilmente. Para los procesos predefinidos este escenario no se ha considerado, ni menos definido ya que esta fuera del alcance de la tesis. Sin embargo, se asume que todos proyectos poseen valores fijos y conocidos para los *Context Attributes* al principio del proyecto, lo que es un prerrequisito para aplicar ambas estrategias de adaptación presentadas; aunque en general no es una limitación fuerte ya que las SSE en general realizan proyectos cortos y no es frecuente que cambie el contexto durante la realización de dichos proyectos. Por otro lado, al definir sólo cinco contextos inesperados provenientes de escenarios reales y posibles, el grado de representatividad de ellos no ha sido validado. Si bien algunas configuraciones podrían ser más

probables o factibles en la práctica, con el propósito de comparar estrategias de adaptación de procesos, se ha considerado un número de contextos que cubran de forma suficiente los escenarios reales y que tengan la potencialidad de ser razonables para un proyecto típico en la SSE.

Sería interesante evaluar según rangos de similitud a los contextos de los procesos predefinidos, con respecto a todos los contextos posibles para cada empresa, i.e., 106 para Rhiscom y 2304 para Mobius. Esto permitiría distinguir si a mayor similitud entre contextos, se logra una menor cantidad de tareas extras y tareas faltantes.

Para definir los procesos predefinidos y sus contextos, fue necesario recopilar la experiencia de los miembros de ADAPTE que apoyaron a las empresas a formalizar sus procesos. Para los contextos inesperados, se eligieron escenarios que fuesen similares y a la vez realistas, comparados a los más tipicos para evaluar el comportamiento de las estrategias. Lo anterior se debe a que para ambas empresas, no existen registros del uso del proceso o del contexto utilizado en cada proyecto, lo cuál es un elemento que aún no se enecuentra disponible. Sin embargo, sería posible obtener datos a través de entrevistas con empleados de las empresas. Con datos como estos sería posible presentar análisis más acabados.

Capítulo 6

Conclusiones y trabajo futuro

Durante el transcurso de esta tesis se ha usado la comparación entre modelos para evaluar el beneficio de usar una estrategia de adaptación automática de procesos, o bien un conjunto de procesos predefinidos. La estrategia propuesta se ha validado considerando dos SSE chilenas: Rhiscom y Mobius. Por un lado, el proceso organizacional de Rhiscom posee 36 tareas y el de Mobius posee 106, mientras que la cantidad de elementos variables es proporcionalmente similar, i.e., 35, 11 % de las tareas son variables en Rhisom y 39,62 % lo son para Mobius. Bajo estos términos, Mobius posee un proceso organizacional más detallado que Rhiscom.

Para cada SSE, el experimento realizado comprendió definir ocho contextos: tres correspondientes a los pares contexto-proceso o contextos de plantillas de los procesos predefinidos y cinco que representan proyectos reales o también llamados contextos inesperados. Para cada uno de los contextos inesperados se efectuaron las adaptaciones con ambas estrategias; mientras que por construcción la adaptación automática obtiene los procesos óptimos, los procesos predefinidos seleccionan un proceso según la mayor similitud. La similitud es la suma de los pesos correspondientes a los *Context Attributes Values* que coinciden para cada *Context Attribute* entre un contexto inesperado y un contexto de plantillas. Los pesos de cada *Context Attribute* han sido obtenidos al inspeccionar los procesos organizacionales de cada empresa e inspeccionando las tareas afectadas por un determinado *Context Attribute*.

Posteriormente se mide el beneficio de utilizar una estrategia con respecto a otra según la productividad y calidad del proceso resultante. Con este fin en mente son definidas las métricas de tareas extras y tareas faltantes, respectivamente. Al obtener los procesos con cada estrategia, se comparan de forma correspondiente según dichas métricas. Para refinar los resultados de los procesos predefinidos, se ha efectuado un *merge* entre los dos procesos de plantillas que posean el más alto valor de similitud con respecto al contexto inesperado, siempre y cuando el valor absoluto de la diferencia entre sus similitudes esté bajo un umbral. Lo anterior responde a la intuición de que si dos similitudes son parecidas, los respectivos procesos también lo son y el *merge* entre ellos mitigaría la cantidad de tareas extras y faltantes.

Los resultados de comparar ambas estrategias muestran que para un proyecto, las plantillas

seleccionadas mediante los procesos predefinidos son aproximadas, mientras que la adaptación automática otorga resultados óptimos en presencia de cualquier contexto; esto último es una presunción con amplios fundamentos (Sección 5.4). Aunque es necesaria una muestra mayor para efectuar conclusiones más fuertes, las plantillas definidas para Rhiscom logran resultados más cercanos a la adaptación automática debido a las características de su proceso organizacional: baja cantidad de elementos variables y condiciones de variabilidad fáciles de cumplir. Por otro lado, las plantillas definidas para Mobius logran resultados con una mayor cantidad de tareas extras y tareas faltantes debido a que su proceso organizacional posee una granularidad fina con respecto a las tareas, una gran cantidad de elementos variables y condiciones de variabilidad bien definidas.

Nuevamente, bajo la premisa de que los procesos obtenidos mediante la adaptación automática son óptimos para cualquier contexto, es posible confirmar

Hipótesis 1: *Los procesos predefinidos serán igual de adecuados a la adaptación automática, cuando los contextos inesperados correspondan exactamente a los procesos predefinidos.*

En primer lugar, esta hipótesis por construcción se cumple. Sin embargo, en el caso de Rhiscom existen contextos inesperados, e.g., D, F, G y H, donde los procesos predefinidos pueden ser igual de adecuados (Cuadro 5.7). En el caso de Mobius y según los datos considerados, no se presentan contextos inesperados donde los procesos predefinidos puedan ser igualmente adecuados que la adaptación automática (Cuadro 5.11).

Hipótesis 2: *Una variación en el contexto impacta tanto en productividad y calidad al proceso resultante.*

Según los datos considerados para Rhiscom, no es posible confirmar esta hipótesis. Como lo muestra la Tabla 5.6, la similitud entre el contexto C y el contexto inesperado D es cero, pero también lo es la cantidad de tareas extras y tareas faltantes (Cuadro 5.7). Según los datos considerados para Mobius, la hipótesis es posible confirmarla, ya que en cada caso se muestra una variación de tareas extras o faltantes (Cuadro 5.11).

Hipótesis 3: *Al realizar un* merge *entre los procesos predefinidos de mayor similitud, bajo un umbral definido, se obtendrá un proceso más similar al óptimo.*

Con los datos obtenidos, no es posible confirmar la última hipótesis. Sin embargo, los datos de Rhiscom (Cuadros 5.7 y 5.8) y de Mobius (Cuadros 5.11 y 5.12), muestran que el *merge* disminuye la cantidad de tareas faltantes a un costo de tareas extras.

Un aspecto que ha afectado los resultados, es como estan formadas las reglas de variabilidad dentro de los procesos. En el caso de Rhiscom, como los puntos de variabilidad poseen condiciones que fácilmente pueden cumplirse, la cantidad de de tareas extras y faltantes es baja. Según lo observado la mayor cantidad de tareas extras es cuatro, que es el 30,77 % de los elementos variables para esta empresa. En cambio para Mobius, las condiciones en los puntos de variabilidad requieren de uno o dos valores específicos. Esto se traduce en que

cantidad de tareas diferentes va desde 11 hasta 30, que corresponden al 26,19 % y 71,42 % del total de elementos variables, respectivamente.

Las ventajas de la adaptación automática contemplan la completa precisión del proceso adaptado para todo contexto, siendo utilizable para cualquier empresa que esté en vías de incrementar su madurez. Sin embargo, comprende una alta complejidad de construcción e implementación, conocimientos altamente sofisticados para una SSE. Con el objetivo de promover su uso en las SSE chilenas, se ha aumentado la usabilidad para definir las reglas de variabilidad y efectuar las adaptaciones [76], así como también de proveer una infraestructura que permita la definición y evolución de los modelos que interactúan en estos procedimientos mediante un megamodelo [10].

La adaptación basada en plantillas es una estrategia frecuentemente utilizada ya sea por su más fácil definición o por cumplir con las necesidades de la empresa: obtener rápidamente un proceso que corresponda a uno de los tipos de proyectos más comunes. La comparación realizada sugiere que es una estrategia subóptima para la obtención de procesos, más aún su uso provocaría disminuciones tanto en productividad como en calidad en los proyectos, pero es un primer paso en la formalización.

La metodología utilizada ha sido satisfactoria para cumplir con los objetivos planteados:

- Se ha definido la arquitectura correspondiente a la estrategia basada en plantillas.
- Se ha construido un seleccionador de procesos en base la mayor similitud entre contextos.
- Se ha construido un comparador de procesos, que ha permitido establecer la cantidad de tareas extras y tareas faltantes entre ellos.
- Se ha refinado la obtención del proceso para la estrategia de procesos predefinidos, al obtener el *merge* entre dos plantillas.
- Se ha reconocido el caso que obtiene una menor cantidad de tareas extras y tareas faltantes: una gruesa granularidad con respecto a las tareas, una baja cantidad de elementos variables y condiciones de variabilidad muy flexibles.
- Se han identificado las amenazas a la validez de este trabajo.

6.1. Contribuciones

Las contribuciones de este trabajo son:

- Una metodología generalizable a otras SSE, ya sea para evaluar el proceso de software utilizado o para motivar el uso de procesos de software.
- Comprobar la factibilidad de comparar dos estrategias de adaptación de procesos de software mediante una comparación entre modelos.

- Otorgar datos objetivos sobre el beneficio—con respecto a *Process Elements*—de comparar dos estrategias de adaptación de procesos de software.

Utilizando la metodología presentada en esta tesis, fue posible comparar dos estrategias de adaptación de procesos: automática y basada en plantillas, correspondientes a la adaptación propuesta por ADAPTE y a un conjunto de procesos predefinidos. Asimismo, fue posible medir el beneficio según la cantidad de *Process Elements*, lo que requirió contar con los procesos explícitos. En el caso de que la empresa no disponga de su proceso organizacional definido y formalizado, entonces es posible generar los procesos correspondientes a los proyectos más comunes y desde ellos abstraer el proceso general y su variabilidad; lo que permite evitar las anomalías presentadas por los procesos adaptados de Rhiscom, donde sus procesos adaptados se diferenciaban por una cantidad pequeña de tareas extras o tareas faltantes.

Ya han sido documentados los resultados de la adopción de un proceso organizacional con respecto a su uso [9, 73]. Sin embargo, dado que formalizar una estrategia de adaptación es un esfuerzo muy grande y en presencia de la limitada capacidad técnica que posee una SSE, aún no es prioridad adoptar una metodología con una infraestructura tan compleja. En el caso particular de ADAPTE, para formalizar un proceso de software en general participan de la formalización tres expertos provenientes de la academia y cuatro personas de la empresa, durante 8 sesiones de cuatro horas; con un total de 128 horas-persona para la SSE como mínimo. Se presume que las SSE chilenas no han dirigido sus esfuerzos en esta tarea, especialmente cuando el retorno de la inversión es poco claro.

Al mostrar que una adaptación basada en plantillas, más precisamente los procesos predefinidos, poseen un desempeño subóptimo con respecto a la adaptación automática propuesta por ADAPTE, se indica en términos cuantificables y generalizables el beneficio de formalizar un proceso organizacional de software con sus variabilidades. Esto permite que las SSE chilenas basen la decisión de formalizar sus procesos y utilizar la estrategia de adaptación automática en hechos objetivos.

6.2. Trabajo futuro

En esta tesis, se ha considerado el beneficio en función de las tareas extras y tareas faltantes para establecer si es adecuada una plantilla seleccionada desde los procesos predefinidos, comparada con los procesos adaptados automáticamente. Es de interés ampliar esta noción, al identificar y analizar otras métricas que permitan llevar a cabo mejoras en los procesos, o bien generar otras con respecto a los mismos *Process Elements*, la ejecución del proceso o su adopción.

Es de interés aplicar el experimento realizado a todos los contextos inesperados disponibles, lo que permitiría establecer las características de los contextos y como se relacionan al número de tareas extras y tareas faltantes. Asimismo, evaluar y recopilar datos directamente de la

industria mediante entrevistas sería una excelente manera de validar algunas suposiciones realizadas en el transcurso de la tesis.

Hasta ahora se ha evaluado el beneficio entre las estrategias de adaptación con respecto a la cantidad de tareas extras y tareas faltantes. Es de interés considerar otras unidades de medida que permitan calcular el retorno de la inversión, y que además reflejen de una forma más cercana el ambiente organizacional de una empresa, e.g., las horas-persona. Esto es posible lograrlo al asociar cada tarea con la cantidad de tiempo esperada para su ejecución. Esto permitiría calcular el tiempo extra o tiempo faltante que toma un proyecto, además de indicar el tiempo ahorrado al ocupar la adaptación automática. Para establecer la cantidad de horas-persona por tarea, se necesita de un conjunto de ejecuciones que permitan estimar estos valores o bien la experiencia de una persona encargada de dichas estimaciones como un jefe de proyectos.

Este problema que está fuera del alcance de esta tesis resulta de interés: analizar, identificar y generar métricas que permitan evaluar estados inconsistentes de un proceso, junto a métodos que restauren dichos procesos de forma automática; o bien extender la adaptación automática para que considere otros *Process Elements*.

Por otro lado, la estrategia basada en plantillas presentada es una opción que brinda resultados ajustados para las SSE que deseen incorporar niveles iniciales de madurez, especialmente cuando la empresa posea tipos de proyectos bien definidos y acotados. Existe la intuición de cuáles son los factores que permiten a una empresa utilizar los procesos predefinidos, e.g., alta granularidad en tareas, pocos elementos variables y condiciones de variabilidad flexibles; pero es de interés obtener una caracterización más acabada con el fin de proponer los procesos predefinidos como un paso inicial en la formalización de procesos de software.

Bibliografía

[1] Marcus Alanen and Ivan Porres. Difference and Union of Models. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *«UML» 2003 - The Unified Modeling Language, Modeling Languages and Applications, 6th International Conference, San Francisco, CA, USA, October 20-24, 2003, Proceedings*, Lecture Notes in Computer Science, pages 2–17, 2003.

[2] Ove Armbrust, Masafumi Katahira, Yuko Miyamoto, Jürgen Münch, Haruka Nakao, and Alexis Ocampo. Scoping Software Process Lines. *Software Process: Improvement and Practice*, 14(3):181–197, 2009.

[3] Asociación Gremial de las empresas chilenas desarrolladoras de software. Sexto Diagnóstico de la Industria Nacional de Software y Servicios. Technical report, GECHS A.G., June 2008.

[4] Xu Bai, LiGuo Huang, and He Zhang. On Scoping Stakeholders and Artifacts in Software Process. In *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, Paderborn, Germany, July 8-9, 2010. Proceedings*, pages 39–51, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] Victor R. Basili and H. Dieter Rombach. Tailoring the Software Process to Project Goals and Environments. In William E. Riddle, Robert M. Balzer, and Kouichi Kishida, editors, *Proceedings, 9th International Conference on Software Engineering, Monterey, California, USA, March 30 - April 2, 1987.*, pages 345–359. ACM Press, 1987.

[6] Victor R. Basili and H. Dieter Rombach. Tailoring the Software Process to Project Goals and Environments. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 345–357, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[7] Victor R. Basili and H. Dieter Rombach. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Trans. Software Eng.*, 14(6):758–773, 1988.

[8] Victor R. Basili and H. Dieter Rombach. The TAME Project: Towards Improvement-oriented Software Environments. *IEEE Trans. Softw. Eng.*, 14(6):758–773, June 1988.

[9] María Cecilia Bastarrica, Gerardo Matturro, Romain Robbes, Luis Silvestre, and René Vidal. How does Quality of Formalized Software Processes Affect Adoption? In Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos,

Haralambos Mouratidis, and Jennifer Horkoff, editors, *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, pages 226–240. Springer International Publishing, 2014.

[10] María Cecilia Bastarrica, Jocelyn Simmonds, and Luis Silvestre. Using Megamodeling to Improve Industrial Adoption of Complex MDE Solutions. In *6th International Workshop on Modeling in Software Engineering, MiSE 2014, Hyderabad, India, June 2-3, 2014*, pages 31–36, 2014.

[11] Sami Beydeda, Matthias Book, and Volker Gruhn, editors. *Model-Driven Software Development*. Springer, 2005.

[12] Jean Bézivin. On the unification power of models. *Software and System Modeling*, 4(2):171–188, 2005.

[13] Jean Bézivin. Model Driven Engineering: An emerging technical space. In *Generative and transformational techniques in software engineering*, pages 36–64. Springer, 2006.

[14] Pedro Borges, Paula Monteiro, and Ricardo Jorge Machado. Mapping RUP Roles to Small Software Development Teams. In *Software Quality. Process Automation in Software Development - 4th International Conference, SWQD 2012, Vienna, Austria, January 17-19, 2012. Proceedings*, pages 59–70, 2012.

[15] Lionel Briand, Khaled El Emam, and Walcélio L. Melo. Ainsi: An inductive method for software process improvement: Concrete steps and guidelines. Technical report, University of Maryland, College Park, MD, USA, 1995.

[16] Petra Brosch, Gerti Kappel, Philip Langer, Martina Seidl, Konrad Wieland, and Manuel Wimmer. An Introduction to Model Versioning. In Marco Bernardo, Vittorio Cortellessa, and Alfonso Pierantonio, editors, *Formal Methods for Model-Driven Engineering - 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2012, Bertinoro, Italy, June 18-23, 2012. Advanced Lectures*, volume 7320 of *Lecture Notes in Computer Science*, pages 336–398. Springer, 2012.

[17] Cédric Brun and Alfonso Pierantonio. Model Differences in the Eclipse Modeling Framework. *UPGRADE, The European Journal for the Informatics Professional*, 9(2):29–34, 2008.

[18] David W. Bustard and Frank Keenan. Strategies for Systems Analysis: Groundwork for Process Tailoring. In *12th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS 2005), 4-7 April 2005, Greenbelt, MD, USA*, pages 357–362. IEEE Computer Society, 2005.

[19] Antonio Cicchetti and Federico Ciccozzi. Towards a Novel Model Versioning Approach Based on the Separation Between Linguistic and Ontological Aspects. In *Proceedings of the Workshop on Models and Evolution co-located with ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013), Miami, Florida, USA, October 1, 2013.*, pages 60–69, 2013.

[20] Antonio Cicchetti, Davide Di Ruscio, and Alfonso Pierantonio. A Metamodel Independent Approach to Difference Representation. *Journal of Object Technology*, 6(9):165–185, 2007.

[21] Paul C. Clements and Linda M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[22] CMMI Product Team. CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1 Staged Representation (CMU/SEI-2002-TR-012, ESC-TR-2002-012). Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, March 2002.

[23] Alistair Cockburn. *Crystal Clear, a Human-powered Methodology for Small Teams*. Addison-Wesley Professional, first edition, 2004.

[24] European Commission. *The new SME definition*. European Commission, May 2003. Available at `http://ec.europa.eu/enterprise/policies/sme/files/sme_definition/sme_user_guide_en.pdf`.

[25] Kieran Conboy and Brian Fitzgerald. Method and developer characteristics for effective agile method tailoring: A study of XP expert opinion. *ACM Trans. Softw. Eng. Methodol.*, 20(1):2:1–2:30, July 2010.

[26] Reidar Conradi, Christer Fernström, and Alfonso Fuggetta. A Conceptual Framework for Evolving Software Processes. *SIGSOFT Softw. Eng. Notes*, 18(4):26–35, October 1993.

[27] Daniel Dias de Carvalho, Larissa Fernandes Chagas, Adailton Magalhães Lima, and Carla Alessandra Lima Reis. Software Process Lines: A Systematic Literature Review. In *Software Process Improvement and Capability Determination - 14th International Conference, SPICE 2014, Vilnius, Lithuania, November 4-6, 2014, Proceedings*, volume 477 of *Communications in Computer and Information Science*, pages 118–130. Springer International Publishing, 2014.

[28] Eclipse Foundation. Eclipse Modeling Framework, December 2013.

[29] Eclipse Foundation. Eclipse Process Framework, December 2013.

[30] Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Fundamental Theory for Typed Attributed Graph Transformation. In H. Ehrig, G. Engels, F. Parisi-Presicce, and G. Rozenberg, editors, *ICGT*, volume 3256 of *Lecture Notes in Computer Science*, pages 161–177. Springer, 2004.

[31] Mark Ginsberg and Lauren Quinn. Process Tailoring and the the Software Capability Maturity Model. Technical report, CMU/SEI-94-TR-024, Software Engineering Institute, November 1995.

[32] Girschick, Martin and Darmstadt, Tu. Difference Detection and Visualization in UML Class Diagrams. Technical report, TU Darmstadt, 2006.

[33] Hassan Gomaa, Larry Kerschberg, and Ghulam A. Farrukh. Domain Modeling of Software Process Models. In *6th International Conference on Engineering of Complex Computer Systems (ICECCS 2000), 11-15 September 2000, Tokyo, Japan*, pages 50–60, 2000.

[34] Geir K. Hanssen, Hans Westerheim, and Finn Olav Bjørnson. Tailoring RUP to a Defined Project Type: A Case Study. In Frank Bomarius and Seija Komi-Sirviö, editors, *Product Focused Software Process Improvement*, volume 3547 of *Lecture Notes in Computer Science*, pages 314–327. Springer Berlin Heidelberg, 2005.

[35] Julio Ariel Hurtado Alegría. *A Meta-Process for defining adaptable software processes*. PhD thesis, Universidad de Chile, Santiago, Chile, Agosto 2012.

[36] Julio Ariel Hurtado Alegría and María Cecilia Bastarrica. Building Software Process Lines with CASPER. In *2012 International Conference on Software and System Process, ICSSP 2012, Zurich, Switzerland, June 2-3, 2012*, ICSSP '12, pages 170–179, Piscataway, NJ, USA, 2012. IEEE Press.

[37] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, and Alexandre Bergel. AVISPA: A Tool for Analyzing Software Process Models. *Journal of Software: Evolution and Process*, 26(4):434–450, 2014.

[38] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Sergio F. Ochoa, and Jocelyn Simmonds. MDE software process lines in small companies. *Journal of Systems and Software*, 86(5):1153–1171, 2013.

[39] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Alcides Quispe, and Sergio F. Ochoa. An MDE approach to software process tailoring. In *International Conference on Software and Systems Process, ICSSP 2011, Honolulu, HI, USA, May 21-22, 2011, Proceedings*, ICSSP '11, pages 43–52, New York, NY, USA, 2011. ACM.

[40] Julio Ariel Hurtado Alegría, María Cecilia Bastarrica, Alcides Quispe, and Sergio F. Ochoa. MDE-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26(4):386–403, 2014.

[41] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The unified software development process - the complete guide to the unified process from the original designers*. Addison-Wesley object technology series. Addison-Wesley, 1999.

[42] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. *Sci. Comput. Program.*, 72(1-2):31–39, 2008.

[43] Georg Kalus and Marco Kuhrmann. Criteria for Software Process Tailoring: A Systematic Review. In *International Conference on Software and System Process, ICSSP '13, San Francisco, CA, USA, May 18-19, 2013*, pages 171–180, 2013.

[44] Gene Kelly. Barriers to Adoption of the CMMI Process Model in Small Settings. Technical report, Carnegie Mellon University, Pittsburgh, 2006.

[45] Udo Kelter, Jürgen Wehren, and Jörg Niere. A Generic Difference Algorithm for UML

Models. In *Software Engineering 2005, Fachtagung des GI-Fachbereichs Softwaretechnik, 8.-11.3.2005 in Essen*, volume 64 of *LNI*, pages 105–116. GI, 2005.

[46] Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[47] Dimitrios S. Kolovos, Richard F. Paige, and Fiona Polack. Merging Models with the Epsilon Merging Language (EML). In *Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, Proceedings*, pages 215–229, 2006.

[48] Dimitrios S. Kolovos, Davide Di Ruscio, Richard F. Paige, and Alfonso Pierantonio. Different Models for Model Matching: An analysis of approaches to support model differencing. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models*, CVSM '09, pages 1–6. IEEE Computer Society, May 2009.

[49] Robert E. Kraut and Lynn A. Streeter. Coordination in Software Development. *Commun. ACM*, 38(3):69–81, March 1995.

[50] Philippe Kruchten. *The Rational Unified Process: An Introduction, Second Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[51] Marco Kuhrmann, Georg Kalus, and Manuel Then. The Process Enactment Tool Framework-Transformation of Software Process Models to Prepare Enactment. *Sci. Comput. Program.*, 79:172–188, January 2014.

[52] Yuehua Lin, Jeff Gray, and Frédéric Jouault. DSMDiff: A Differentiation Tool for Domain-Specific Models, 2007.

[53] Mark Lycett, Robert D. Macredie, Chaitali Patel, and Ray J. Paul. Migrating Agile Methods to Standardized Development Practice. *IEEE Computer*, 36(6):79–85, June 2003.

[54] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, USA, 2001.

[55] Tom Mens. A State-of-the-Art Survey on Software Merging. *IEEE Trans. Softw. Eng.*, 28(5):449–462, May 2002.

[56] Tom Mens and Pieter Van Gorp. A Taxonomy of Model Transformation. *Electron. Notes Theor. Comput. Sci.*, 152:125–142, March 2006.

[57] Jürgen Münch, Ove Armbrust, Martin Kowalczyk, and Martín Soto. *Software Process Definition and Management.* The Fraunhofer IESE Series on Software and Systems Engineering. Springer, 2012.

[58] Shiva Nejati, Mehrdad Sabetzadeh, Marsha Chechik, Steve Easterbrook, and Pamela Zave. Matching and Merging of Statecharts Specifications. In *29th International Confe-*

rence on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007, pages 54–64, Washington, DC, USA, 2007. IEEE Computer Society.

[59] Dirk Ohst, Michael Welle, and Udo Kelter. Differences between versions of UML diagrams. In *Proceedings of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conference, ESEC/FSE 2003, Helsinki, Finland, September 1-5, 2003*, pages 227–236. ACM, 2003.

[60] OMG. MOF 2.0/XMI Mapping, Version 2.1.1, December 2007.

[61] OMG. Software & Systems Process Engineering Metamodel Specification (SPEM). Technical report, Object Management Group, April 2008.

[62] OMG. Meta Object Facility (MOF) Core Specification. Technical report, Object Management Group, April 2014.

[63] Daniel Ortega, Luis Silvestre, María Cecilia Bastarrica, and Sergio F. Ochoa. A Tool for Modeling Software Development Contexts in Small Software Organizations. In *31st International Conference of the Chilean Computer Science Society, SCCC 2012, Valparaíso, Chile, November 12-16, 2012*, pages 29–35, Los Alamitos, CA, USA, 2012. IEEE Computer Society.

[64] Leon J. Osterweil. Software processes are software too. In *ICSE '87: Proceedings of the 9th International Conference on Software Engineering*, pages 2–13, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[65] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet: Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 1024–1025. AAAI Press, 2004.

[66] Oscar Pedreira, Mario Piattini, Miguel R. Luaces, and Nieves R. Brisaboa. A systematic review of software process tailoring. *SIGSOFT Softw. Eng. Notes*, 32(3):1–6, May 2007.

[67] Minna Pikkarainen and Annukka Mäntyniemi. An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies. In *In Proceedings of Software Process Improvement and Capability determination 2006 Conference*. SPICE Conference, May 2006.

[68] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[69] Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10(4):334–350, December 2001.

[70] Jolita Ralyté, Rébecca Deneckère, and Colette Rolland. Towards a Generic Model for Situational Method Engineering. In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering*, CAiSE'03, pages 95–110, Berlin, Heidelberg, 2003. Springer-Verlag.

[71] Raghu Reddy, Robert France, Franck Fleuery, and Benoit Baudry. Model composition - A Signature-Based Approach. In *Proceedings Aspect Oriented Modeling workshop held with MODELS/UML 2005, Montego*, page 2, 2005.

[72] Louis Rose, Esther Guerra, Juan de Lara, Anne Etien, Dimitris S. Kolovos, and Richard Paige. Genericity for Model Management Operations. *Software and Systems Modeling*, 12(1):201–219, 2013.

[73] Pablo Ruiz, Alcides Quispe, María Cecilia Bastarrica, and Julio Ariel Hurtado Alegría. Formalizing the software process in small companies. *8CCC, Colombia (August 2013)*, 2012.

[74] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[75] Douglas C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.

[76] Luis Silvestre, María Cecilia Bastarrica, and Sergio F. Ochoa. A Model-based Tool for Generating Software Process Model Tailoring Transformations. In *MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, pages 533–540, Lisbon, Portugal, January 2014.

[77] Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using Benchmarking to Advance Research: A Challenge to Software Engineering. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.

[78] Borislava I. Simidchieva, Lori A. Clarke, and Leon J. Osterweil. Representing Process Variation with a Process Family. In Qing Wang, Dietmar Pfahl, and David M. Raffo, editors, *Software Process Dynamics and Agility, International Conference on Software Process, ICSP 2007, Minneapolis, MN, USA, May 19-20, 2007, Proceedings*, volume 4470 of *Lecture Notes in Computer Science*, pages 109–120. Springer Berlin Heidelberg, 2007.

[79] Ian Sommerville. *Software Engineering.* Addison-Wesley, Harlow, England, 9th edition, 2011.

[80] Stanley M. Sutton Jr. and Leon J. Osterweil. Product families and process families. In *10th International Software Process Workshop (ISPW '96), June 17-19, 1996, Dijon, France*, pages 109–111, 1996.

[81] Christoph Treude, Stefan Berlik, Sven Wenzel, and Udo Kelter. Difference computation of large models. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC-FSE '07, pages 295–304, New York, NY, USA, 2007. ACM.

[82] Mark van den Brand, Zvezdan Protić, and Tom Verhoeff. Generic Tool for Visualization of Model Differences. In *Proceedings of the 1st International Workshop on Model*

Comparison in Practice, IWMCP '10, pages 66–75, New York, NY, USA, 2010. ACM.

[83] Mark van den Brand, Zvezdan Protić, and Tom Verhoeff. RCVDiff - a stand-alone tool for representation, calculation and visualization of model differences. In *Conference Paper : ME 2010 - International Workshop on Models and Evolution co-located with ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems*, MoDELS'06. ACM, 2011.

[84] Arie van Deursen, Eelco Visser, and Jos Warmer. Model-Driven Software Evolution: A Research Agenda. In *CSMR Workshop on Model-Driven Software Evolution (MoDSE 2007)*, pages 41–49, March 2007.

[85] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-Based Integration of Information - A Survey of Existing Approaches. In *In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, Seattle, WA, 2001.

[86] Hironori Washizaki. Building Software Process Line Architectures from Bottom Up. In Jürgen Münch and Matias Vierimaa, editors, *Product-Focused Software Process Improvement, 7th International Conference, PROFES 2006, Amsterdam, The Netherlands, June 12-14, 2006, Proceedings*, Lecture Notes in Computer Science, pages 415–421, 2006.

[87] Zhenchang Xing and Eleni Stroulia. UMLDiff: an algorithm for object-oriented design differencing. In *20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), November 7-11, 2005, Long Beach, CA, USA*, pages 54–65. ACM, 2005.

[88] Dave Zubrow. Current trends in adoption of the CMMI® Product Suite. In *27th International Computer Software and Applications Conference (COMPSAC 2003): Design and Assessment of Trustworthy Software-Based Systems, 3-6 November 2003, Dallas, TX, USA, Proceedings*, pages 126–129. IEEE, 2003.