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

Modelador de contexto e interfaz gráfica para evaluación de procesos híbridos de desarrollo de software

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

Daniel Nicolás González Calderón

PROFESORA GUÍA:
María Cecilia Bastarrica Piñeyro
MIEMBROS DE LA COMISIÓN:
Éric Tanter
Juan Arriagada Cancino

SANTIAGO DE CHILE
2024

Resumen

La creciente complejidad en el desarrollo de software ha llevado a la adopción de metodologías híbridas, combinando enfoques ágiles y tradicionales para adaptarse a la dinámica y requisitos específicos de cada proyecto. Sin embargo, personalizar estas metodologías para proyectos específicos, conocido como tailoring, presenta un desafío significativo debido a la necesidad de equilibrar factores como tiempos de entrega, calidad del software y recursos disponibles.

Este trabajo se centra en DynaTail, un método que permite la personalización dinámica de metodologías híbridas de desarrollo de software adaptadas al contexto específico de cada proyecto. Para este trabajo, se buscó implementar una base sobre la cuál se pueda ejecutar este método, manejando sus distintos componentes y permitiendo su futura expansión. La solución propuesta, basada en Model-driven engineering (MDE), utiliza modelos para ofrecer un enfoque sistemático y flexible para la adaptación de procesos de desarrollo, asegurando que estén alineados con las necesidades específicas del proyecto.

La implementación incluye un modelador de contexto que permite definir y generar modelos de contexto de proyectos, esenciales para la personalización del proceso de desarrollo. Además, la interfaz gráfica integra diferentes componentes de DynaTail, facilitando la visualización y evaluación de los procesos híbridos de desarrollo. Esta herramienta proporciona una base sólida para la futura expansión y completa implementación del método DynaTail, contribuyendo de manera importante a la gestión de procesos de desarrollo de software en entornos variados y complejos.

Agradecimientos

En primer lugar, quiero agradecer a mi profesora guía, Cecilia Bastarrica, y al profesor Luis Silvestre por su orientación y apoyo en cada etapa del desarrollo de este trabajo.

Además agradezco a todos mis amigos, cuya compañía, apoyo y motivación fueron pilares fundamentales a lo largo de este camino académico.

A mi familia, quiero expresar mi profundo agradecimiento por su constante respaldo a lo largo de toda mi carrera, que a pesar de las ausencias prolongadas y las demandas académicas, siempre me brindaron su comprensión, amor y aliento.

Asimismo, deseo agradecer a mis jefes en Cardda, Sebastián Hernandez y Cristóbal Grinbergs, por su comprensión y apoyo durante los años en los que trabajé en la empresa mientras cursaba mis estudios universitarios. La flexibilidad y el entendimiento que tuvieron fue clave para enfrentar los desafíos tanto académicos como laborales durante este período.

Por último, me gustaría agradecer a mi gato Shiro por su compañía y cariño a lo largo de este año.

Tabla de contenido

1. Introducción

1.1. Contexto

El desarrollo de software es una actividad intrínsecamente compleja, donde la coordinación de diversos factores, tales como los requisitos del cliente, las tecnologías disponibles, los recursos humanos y materiales, los plazos y los presupuestos, desempeña un papel crucial. Para gestionar esta complejidad, existen diferentes metodologías de desarrollo de software que definen las fases, las actividades, las herramientas y los roles involucrados en el ciclo de vida del software, buscando ofrecer estructuras y enfoques para guiar el proceso de manera efectiva.

En este ámbito, la gestión de proyectos se enfrenta a desafíos complejos, dados los diversos factores que influyen en el proceso. Las metodologías tradicionales, como el modelo en cascada, han destacado por su enfoque riguroso y planificación detallada de cada etapa del desarrollo de software. Este método es apropiado para proyectos con requisitos estables y bien definidos, donde los cambios son mínimos y las entregas parciales no son una necesidad inmediata. Sin embargo, su rigidez puede convertirse en una limitación cuando se enfrentan a proyectos más dinámicos que requieren adaptación continua.

Por otro lado, las metodologías ágiles, como Scrum, han ganado popularidad al enfocarse en la entrega rápida y frecuente de valor al cliente. Estas metodologías son ideales para proyectos con requisitos cambiantes y dinámicos, donde la adaptabilidad y la capacidad de respuesta al cambio son esenciales. A pesar de sus ventajas, también tienen sus limitaciones, especialmente en proyectos que requieren una planificación más detallada y una estructura más definida.

Esta diversidad de proyectos y sus requisitos específicos ha llevado al surgimiento de metodologías híbridas en los últimos años[23]. La combinación de enfoques tradicionales y ágiles ha permitido adaptarse al contexto específico de cada proyecto, proporcionando flexibilidad y personalización.

Un ejemplo concreto es la fusión de metodologías en cascada y Scrum, donde se busca aprovechar la estructura y la planificación del primero y la agilidad del segundo.

En la fase inicial de este enfoque, se adopta el modelo en cascada para definir claramente el alcance y los requisitos del proyecto. Esta fase implica una planificación detallada y una secuencia de pasos bien definidos, estableciendo una base sólida para el desarrollo posterior, de manera similar a las metodologías tradicionales.

Una vez que se han establecido las bases, la metodología se desplaza hacia la implementación de Scrum para las fases subsiguientes. Scrum trabaja en iteraciones cortas y regulares llamadas sprints, donde se desarrollan y entregan incrementos del producto en intervalos predefinidos. Esta fase ágil permite una mayor adaptabilidad a los cambios de requisitos que puedan surgir en el proyecto y le otorga una capacidad de responder de manera ágil a los desafíos que puedan surgir durante el desarrollo.

La fusión de cascada y Scrum se beneficia de la estructura inicial proporcionada por

el modelo en cascada y de la flexibilidad aportada por Scrum en las etapas posteriores del proyecto. Esto permite una planificación detallada y rigurosa en las fases iniciales, asegurando la claridad en los requisitos, mientras que la implementación ágil permite ajustes continuos y entregas incrementales en respuesta a las necesidades cambiantes del cliente y del entorno del proyecto.

Este enfoque híbrido se vuelve especialmente efectivo en proyectos donde se requiere una combinación de estabilidad y adaptabilidad. Por ejemplo, en proyectos con requisitos bien definidos al principio pero con la posibilidad de cambios a medida que avanza el desarrollo, la fusión de cascada y Scrum ofrece una solución equilibrada.

Así, la coordinación efectiva y la comunicación clara entre las distintas partes del equipo son fundamentales para asegurar que la fusión de cascada y Scrum se traduzca en una entrega exitosa del proyecto[32], combinando la eficiencia de la planificación detallada con la agilidad necesaria para enfrentar los desafíos cambiantes del desarrollo de software.

Dicho esto, la implementación exitosa de esta o cualquier metodología híbrida presenta como problema el requerir una comprensión profunda de ambas metodologías y una gestión cuidadosa de la transición entre ellas.

1.2. Relevancia/Motivación

1.2.1. El uso de metodologías híbridas

Las metodologías de desarrollo de software se definen como una combinación de roles, actividades y productos de trabajo [14]. Definir estas metodologías ha sido reconocido como fundamental para gestionar el desarrollo de manera organizada, permitiendo la planificación, programación y provisión de proyectos de software [26]. Sin embargo, una única metodología no es universalmente aplicable a todos los tipos de proyectos, incluso dentro de una misma organización. Por ejemplo, la experiencia requerida para los desarrolladores puede variar significativamente según la complejidad del producto [6]. De manera similar, un proyecto pequeño podría omitir la construcción de varios productos menores necesarios para proyectos más grandes [39]. La actividad de ajustar la metodología de la empresa a las características particulares del proyecto que se está abordando se denomina tailoring, esto es, el acto de adaptar un proceso para soportar la implementación de un proceso para un contexto particular[10].

Los métodos ágiles abordan el desarrollo de software desde un punto de vista diferente. Proponen una serie de prácticas que el equipo de desarrollo adopta y adapta al proyecto en cuestión. En general, los métodos ágiles se centran en las personas involucradas en el desarrollo de software, son principalmente apropiados para equipos pequeños y promueven la productividad, especialmente en proyectos con alta incertidumbre. Por otro lado, no proponen de forma explícita un soporte sólido para las actividades de gestión de proyectos [31].

Varios estudios han demostrado que, en la práctica, las empresas siguen una combinación de prácticas ágiles y tradicionales para el desarrollo de software. Aunque las empresas grandes tienden a definir metodologías tradicionales de software, el pragmatismo las ha llevado a adoptar algunas prácticas ágiles. Por el contrario, las pequeñas empresas que intentan seguir

una metodología completamente ágil pronto se dan cuenta de que se requiere un proceso más estructurado. Por lo tanto, la mayoría de las empresas tienden a aplicar metodologías híbridas, independientemente de su tamaño [21]. Sin embargo, no es fácil evaluar que la combinación de prácticas ágiles y tradicionales adoptadas sea la más apropiada para el objetivo específico que se busca alcanzar en el proyecto [18, 12].

1.2.2. La personalización de las metodologías

En general, la implementación de metodologías híbridas introduce desafíos adicionales, ya que requiere un mayor nivel de experiencia en la gestión y organización por parte de los desarrolladores y líderes del proyecto, frecuentemente requiriendo ingenieros de procesos para su planificación y ejecución. La optimización de variables críticas para el proyecto, como el tiempo de entrega al cliente o la robustez del software, se convierte en una tarea que demanda un enfoque más meticuloso y especializado. La complejidad inherente a esta actividad, marcada por la interconexión de requisitos del cliente, tecnologías disponibles, recursos humanos y materiales, plazos y presupuestos, requiere enfoques flexibles y adaptables para asegurar la eficiencia y la entrega exitosa de productos de alta calidad.

En este panorama, surge la necesidad de una solución que facilite la personalización y optimización dinámica de las metodologías de desarrollo de proyectos de software.

Es de esta necesidad que surge DynaTail (Dynamic Tailoring for Hybrid Software Processes) [35], un método basado en modelos que aborda la necesidad de adaptabilidad dinámica al contexto específico de cada proyecto, brindando una solución para personalizar y optimizar el proceso de desarrollo de manera dinámica. DynaTail inicia su enfoque desde un modelo de contexto y un proceso inicial, que se ajusta iterativamente para obtener un proceso adaptado al proyecto en función de su contexto específico. Su método se basa en ciclos de tailoring, donde el proceso y el contexto se modifican en cada iteración hasta alcanzar un proceso que maximiza o minimiza el atributo deseado.

Así, la implementación efectiva de DynaTail podría ser clave en el ámbito del desarrollo de software, ya que permitiría a las organizaciones adaptar sus procesos de desarrollo de manera dinámica, respondiendo a las necesidades específicas de cada proyecto y optimizando la entrega de productos de alta calidad en entornos cambiantes. Esto fue validado a través del uso del método, de forma manual, por un usuario de una empresa chilena de software de tamaño mediano, dónde indicó que, si bien el método era apropiado y comprensible para realizar el tailoring, este sólo podría ser adoptado en la industria si hubiese una herramienta de apoyo capaz de implementarlo.

1.3. Objetivos

En este trabajo de título, el objetivo general es abordar los desafíos específicos relacionados con la implementación de DynaTail. Se busca ofrecer una interfaz gráfica intuitiva y un módulo de transformación de modelos de contexto para implementar DynaTail en la industria del desarrollo de software.

1.3.1. Objetivos específicos

Para lograr esto, se buscan lograr los siguientes objetivos específicos:

1. Desarrollar una aplicación capaz de definir y generar modelos de contexto de un proyecto.
2. Desarrollar una aplicación con una interfaz gráfica capaz de llevar a cabo un flujo simulado de DynaTail, con los siguientes requerimientos:
   - Debe ser capaz de recibir y visualizar un proceso en formato BPMN.
   - Visualizar grafos de influencias a partir de su archivo en formato XMI.
   - Permitir seleccionar la característica que se busca optimizar y el tipo de optimización.
   - Poder visualizar el resultado de dicha optimización.

Esta contribución no solo fortalecerá la base teórica y práctica del tailoring sobre las metodologías híbridas de desarrollo de software, sino que también podrá ser extendida para consolidar a DynaTail como una herramienta práctica y eficaz para la industria, permitiendo a organizaciones analizar y mejorar la calidad, productividad y satisfacción en el desarrollo de software de manera continua y dinámica.

Para el módulo de transformación de modelos de contexto, se plantea que este sea capaz de generar un modelo de contexto organizacional, esto es, un modelo con atributos relevantes de la organización que pueden tomar distintos valores, además de valores para estos atributos y así generar un modelo de contexto del proyecto a través de una transformación de estos valores sobre el modelo organizacional (Figura 1). En el área de MDE (Model-Driven Engineering), existen distintos tipos de transformación de modelos. En este caso, al tratarse de una transformación de un modelo que genera otro modelo, se le llama model-to-model transformation, definida como la transformación de los elementos descritos de un modelo a los elementos del modelo objetivo [4]. De esta forma, el modelo resultante podría ser utilizado como input para el tailoring del proceso de desarrollo realizado por DynaTail para la organización en cuestión.

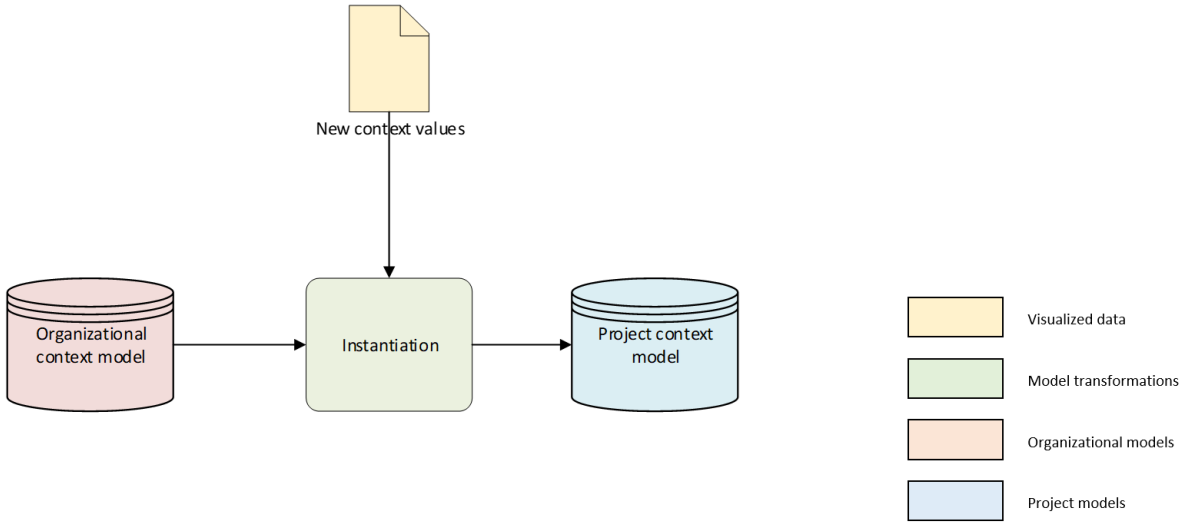


Figura 1: Transformación de modelo de contexto organizacional a un modelo de contexto del proyecto

Cuanto al desarrollo de la interfaz gráfica, se busca que esta interfaz coordine la ejecución de todos los módulos ya desarrollados de DynaTail (Figura 2), permitiendo unificar todo su flujo en una sola plataforma, facilitando así el uso de esta herramienta por parte de los encargados de la organización de un proyecto de software, como un ingeniero de procesos. Esta interfaz gráfica invocaría los distintos módulos con los respectivos valores dados por el usuario, haciendo así que, una vez todos los módulos estén desarrollados, el tailoring pueda ser ejecutado de forma intuitiva, sin la necesidad de cálculos a mano ni análisis de código.

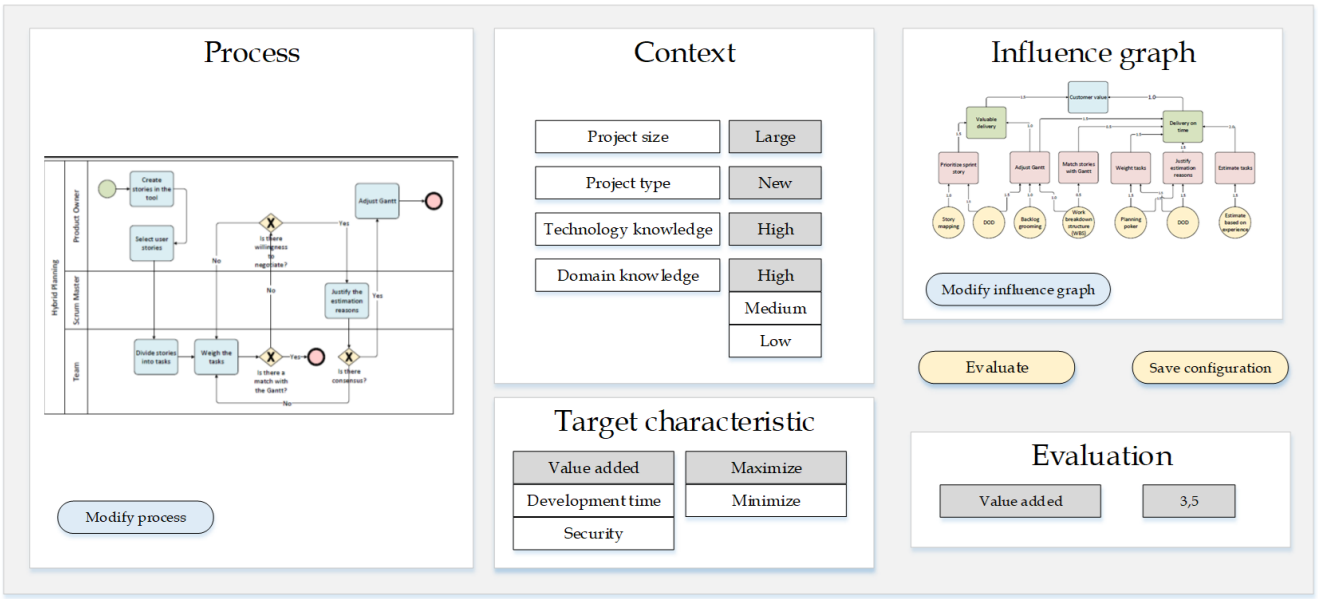


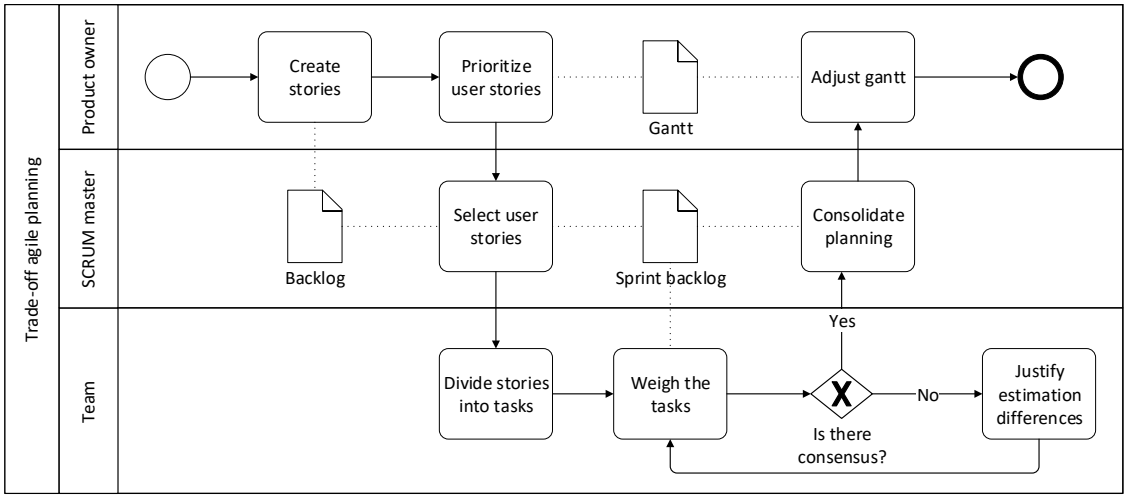
Figura 2: Mockup de las distintas partes de la interfaz gráfica

1.4. Descripción general de la solución

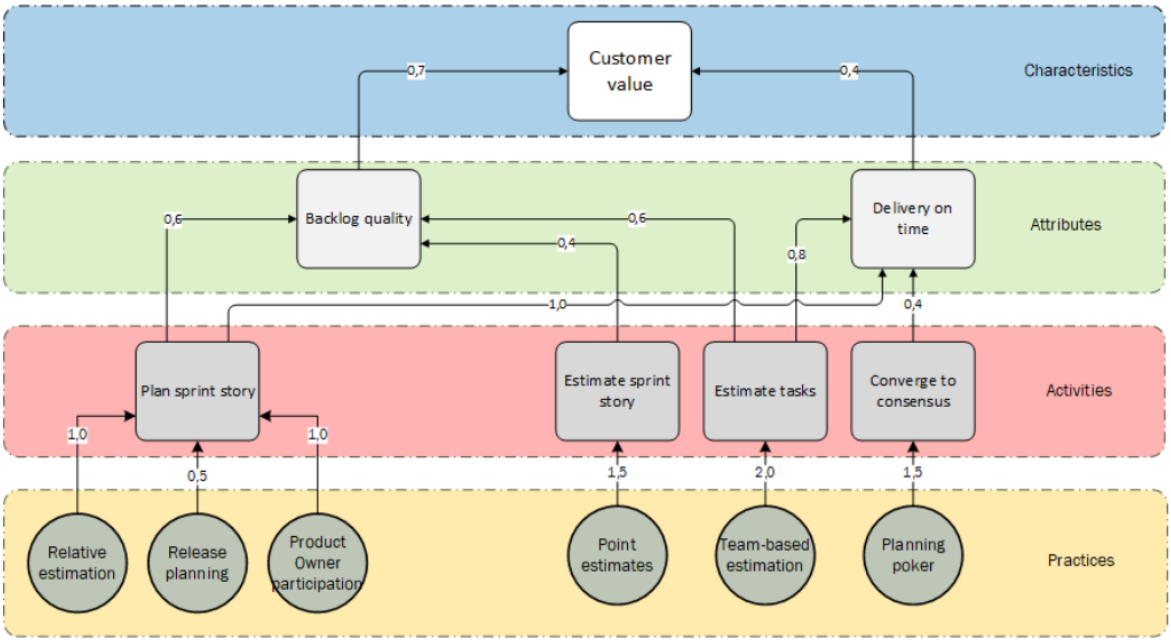
Los módulos de esta herramienta consistirían en el modelo del proceso inicial y adaptado (Figura 3.a), que con la interfaz podría ser visualizado; el modelador del contexto del proyecto implementado en este trabajo de título, generado a partir de un modelo de contexto organizacional; el grafo de influencias indicando los distintos pesos de cada parte del proceso (Figura 3.b); las reglas de tailoring que permiten modificar el proceso en base al contexto del proyecto y finalmente el módulo que realice la evaluación del proceso adaptado [39].

El flujo general para la herramienta sería el siguiente:

1. Se opta por crear un nuevo proyecto o cargar uno antiguo;
2. Se agregan las dimensiones necesarias para el contexto organizacional de la empresa, seleccionando los atributos para instanciar al modelo de contexto del proyecto;
3. Se carga el proceso de desarrollo utilizado por la organización, en formato BPMN, para que este pueda ser visualizado;
4. Se invoca un ejecutable para transformar el proceso en formato BPMN a un modelo de proceso en formato XMI, para su posterior uso;
5. Se visualiza el grafo de influencias utilizado, y se selecciona el atributo que busca ser optimizado;
6. Se selecciona el tipo de optimización (minimizar o maximizar), y se evalúa el proceso;
7. En caso de que el valor evaluado no sea lo suficientemente óptimo, se modifica el proceso o el contexto y se realiza una nueva evaluación.



(a) Proceso de desarrollo híbrido



(b) Grafo de influencias

Figura 3: Ejemplos de visualización de algunos elementos utilizados por DynaTail [35]

2. Estado del Arte

2.1. Metodologías de desarrollo de software

El problema de elegir y adaptar un proceso de desarrollo de software al contexto de cada proyecto no es nuevo, pero tampoco tiene una solución definitiva. Si bien durante muchos años el uso de metodologías tradicionales era la norma, al comienzo del siglo XXI se popularizaron las metodologías ágiles de desarrollo de software [42], debido en gran parte al paso acelerado con el que ambientes de negocios y de tecnologías pasaron a cambiar. Esto hizo con que la flexibilidad de los equipos de desarrollo de software al responder ante cambios en los requerimientos de su sistema pasara a ser uno de los principales factores en el éxito de un proyecto [13], algo que no se daba con metodologías tradicionales de desarrollo.

Sin embargo, luego de un cierto tiempo se empezaron a notar las limitaciones de este tipo de metodología, como una predictibilidad limitada al estimar el esfuerzo requerido para completar ciertos componentes del software, y el posible desvío en la dirección sobre la cuál el proyecto va progresando, a causa de una planificación inicial mínima debido a requerimientos cambiantes [42].

Dadas las limitaciones de ambas estrategias, comúnmente estas son utilizadas en conjunto, permitiendo así que la planificación y evaluación del proceso de desarrollo sean más fáciles. A esta combinación de metodologías se le llama metodología híbrida [42].

Un estudio global respecto a la aplicación de metodologías híbridas en el desarrollo muestra que, a nivel internacional, el uso de estas ocurre en el 76.8 % de las empresas, sin importar su tamaño o el de la industria, indicando también que estas prácticas surgen de la planificación y selección pragmática de procesos, además de su evolución a lo largo del tiempo de acuerdo a las necesidades de la organización [23].

Sin embargo, no cualquier combinación de prácticas es apropiada [30] para la organización y los objetivos de un proyecto, por lo que se hace necesario también una forma de evaluar si una metodología es efectiva para lo que busca lograr. De por sí, la evaluación de cada combinación ya es una tarea desafiante, como ya descubrió Unterkalmsteiner et al. [37]. Además, decidir el grado apropiado de agilidad para la metodología usada también presenta dificultades [7], ya que se deben abordar cuestiones como qué actividades se deben involucrar con cada enfoque y qué práctica es más adecuada para implementar cada actividad. Aunque existen algunas pautas obtenidas empíricamente [36], las características deseadas del proceso y las prácticas disponibles evolucionan con el tiempo, por lo que la elección continua del conjunto apropiado de prácticas debe ajustarse constantemente [18, 24].

Dada esta necesidad de adaptar la forma en la que se desarrolla software, existen diversas propuestas y herramientas que intentan abordar este problema desde diferentes perspectivas y niveles de abstracción.

En la siguiente sección, se explorarán algunos conceptos cruciales para establecer un marco de referencia claro que permita comprender mejor cómo se pueden adaptar eficazmente los procesos de desarrollo de software a las necesidades específicas de cada proyecto.

2.2. Conceptos clave

2.2.1. Modelo

La palabra modelo se refiere a su uso en el área de Model-Driven Engineering (MDE), definida como una abstracción del mundo real que captura una vista, esto es, un modelo describe conceptualmente los aspectos de un sistema que sean relevantes de su punto de vista, con un cierto grado de detalles [11].

2.2.2. Contexto

El contexto se refiere al conjunto de factores internos y externos que influyen en el proyecto, como el tipo de producto a ser desarrollado, los recursos disponibles para llevar a cabo su desarrollo, las herramientas utilizadas y las condiciones ambientales [17].

Para este trabajo de título se utiliza el concepto de modelo de contexto, ya sea organizacional o de proyecto, que se compone de tres elementos principales: dimensiones, atributos y posibles valores de atributos. A continuación, se detallan estos elementos [27]:

- **Dimensiones:** Representan un grupo de atributos contextualmente relacionados. Ejemplos de dimensiones son categorías como el proyecto, el equipo y el producto, cada una con sus propios atributos específicos.
- **Atributos:** Representan a un elemento contextual particular dentro de una dimensión. Por ejemplo, dentro de la dimensión del producto, se pueden tener atributos como la "Complejidad técnica" o la "Importancia de la calidad".
- **Valor atributo:** Expresa un valor específico asumido por un atributo del contexto. Por ejemplo, para el atributo "Complejidad técnica", los posibles valores pueden ser {alta, media, baja}.

2.2.3. Metodología y Proceso

Es importante mencionar la diferencia entre metodología y proceso: mientras el primero se refiere a la forma organizacional en la que se desarrolla el software, el segundo concepto se refiere a un conjunto de pasos parcialmente ordenados que buscan llegar a un objetivo [10].

En el contexto de esta memoria, el proceso será representado por un modelo que conforma con BPMN 2.0 (Business Process Model and Notation) en formato XMI. BPMN es un estándar para la modelización de procesos de negocios, que proporciona un lenguaje gráfico para representar secuencias de actividades y flujos de trabajo de manera comprensible tanto para los participantes técnicos como para los no técnicos. Esto puede ser visualizado en la figura 3(a). Este estándar se utiliza ampliamente para documentar, analizar y automatizar procesos de negocio complejos [1].

Un archivo BPMN describe visualmente las actividades, eventos, decisiones y flujos de control que constituyen un proceso. Los diagramas BPMN son eficaces para mostrar la interacción entre diferentes roles y sistemas, y para identificar oportunidades de mejora o personalización en los procesos. En el marco de DynaTail, por ejemplo, BPMN se utiliza para representar y ajustar los procesos de desarrollo de software híbridos a un contexto específico de un proyecto [35].

Por otro lado, XMI (XML Metadata Interchange) es un estándar basado en XML utilizado para intercambiar metadatos a través de diferentes plataformas y herramientas [41]. En el contexto de BPMN, XMI se utiliza para almacenar y transferir modelos de procesos de manera estandarizada, lo que facilita la interoperabilidad entre diferentes sistemas y herramientas de modelado. XMI permite la serialización de modelos BPMN en un formato que puede ser procesado automáticamente, favoreciendo la integración y automatización de procesos en entornos de desarrollo de software.

2.2.4. Grafo de Influencias

Se trata de un grafo que permite evaluar la calidad del proceso adaptado según una característica objetivo específica, modelado como un "Modelo de Grafo de Influencia". En este modelo, se especifican los atributos que influencian la característica objetivo que se desea mejorar, junto con el peso de esta influencia. Del mismo modo, el conjunto de actividades en el 'Modelo de Proceso' puede influir en cada uno de estos atributos con diferentes pesos. Además, cada actividad puede implementarse con diferentes prácticas, cada una con un peso de influencia distinto [39]. Un ejemplo de este grafo puede ser visualizado gráficamente en la figura 3(b), con sus pesos y distintas partes.

Existe un grafo de influencias diferente para cada característica, y estos son específicos para una organización determinada, ya que las actividades incluidas se relacionan con el 'Modelo de Proceso' y las prácticas son aquellas que se aplican regularmente dentro de la organización. Por lo tanto, los pesos también son específicos de la organización y varían entre -2 y 2, según lo sugerido por Diebold et al. [8], donde -2 indica una influencia altamente negativa y 2 es una influencia altamente positiva. Estos pesos podrían ser ajustados con el tiempo por la organización como consecuencia de resultados empíricos en proyectos anteriores.

2.3. Soluciones existentes

Dicho esto, existen muchas propuestas que mezclan metodologías tradicionales y ágiles, definiendo distintas formas de abordar el proceso del desarrollo de software. Sin embargo, es difícil que un proceso definido de forma general pueda cumplir con los requerimientos de un proyecto u organización específica. Teniendo en consideración que las organizaciones son distintas, y que dentro de una misma organización dos proyectos también pueden ser distintos, una metodología que podría ser aplicada de forma exitosa a un proyecto podría ser un fracaso en otro [29].

En base a esto, se hace necesario realizar un tailoring del proceso de desarrollo para el éxito de cada proyecto. Aún así, la forma en la que este tailoring se realiza, en general, es a través de un acercamiento ad-hoc, sin seguir reglas o guías en su ejecución [29]. Dado esto, se han propuesto tanto métodos para realizar tailoring a procesos ágiles de desarrollo como para procesos híbridos.

Una de estas propuestas para procesos ágiles se llama Situation Context Framework [2], que busca soportar la selección y tailoring de estrategias ágiles dependientes de la situación, utilizando elementos del contexto para indicar la forma en que estos trabajan en conjunto como un todo. Otra de estas propuestas es el Agile Practices Impact Model [9], que busca modelar el impacto de las prácticas ágiles más comunes, pudiendo así ser utilizado para el

tailoring de procesos ágiles.

Cuanto a las propuestas para procesos híbridos, si bien hay estudios para determinar el impacto e importancia de ciertos aspectos de los procesos híbridos, sirviendo así de guía para cualquier tailoring que se realice caso a caso, no existe actualmente una herramienta capaz de realizar este tailoring de forma sistemática y programática [36]. Aún así, estos estudios pueden servir de base para determinar aspectos importantes que una herramienta así deba tener en consideración en su implementación. Uno de estos estudios es el realizado por Vijayasarathy y Butler [38], que determina que los procesos híbridos son en general asociados con características como proyectos de tamaño mediano preocupados por el presupuesto, la alta criticidad del proyecto y que presentan un equipo de desarrollo pequeño. Otro estudio, encabezado por Klünder [19], determina que sólo algunos factores del contexto, como el tamaño del proyecto y el dominio de la aplicación objetivo, influyen significativamente sobre la elección de los métodos a ser utilizados.

Así, como la forma de organización al desarrollar software afecta directamente los objetivos de un proyecto, se torna necesario que las metodologías utilizadas sean acorde a lo que se espera obtener a lo largo del ciclo de vida del software [22]. Dado esto, se debe considerar al contexto como un factor clave para el éxito o el fracaso de la aplicación de alguna metodología de desarrollo en un proyecto de software. Un proceso de desarrollo adecuado al contexto puede facilitar la comunicación, la colaboración, la innovación y la adaptación al cambio. Un proceso de desarrollo inadecuado para un contexto particular puede generar conflictos, retrasos, errores y desperdicios. Por lo tanto, se hace necesario contar con una herramienta como DynaTail (Figura 4), capaz de personalizar los procesos utilizados para cada proyecto de acuerdo a sus necesidades y objetivos, contando con una interfaz gráfica capaz de integrar todos sus componentes, permitiendo visualizar y evaluar el desempeño de los procesos híbridos de desarrollo de software de forma dinámica.

2.4. DynaTail

DynaTail es un método propuesto para el tailoring de metodologías de desarrollo de software, adaptándolas al contexto específico de cada proyecto. Esto se traduce en dos objetivos: 1) permitir la personalización del proceso de organización con tal de adaptarlo al contexto del proyecto en busca de un cierto objetivo. Este tailoring permitiría decidir si es mejor aplicar prácticas ágiles o tradicionales a cada parte de las actividades involucradas en el proceso entero; 2) proveer un framework para probar distintos procesos y modificaciones de contexto con tal de obtener mejores resultados para un proyecto específico [35].

El tailoring requiere de tres elementos: un proceso, un contexto, y un conjunto de reglas de tailoring, y consiste en adaptar el proceso al contexto particular aplicando estas reglas, resultando en un nuevo proceso. Las reglas de tailoring definen los cambios que deben ocurrir en el proceso de acuerdo al contexto particular del proyecto. De manera concreta, esto quiere decir que si un atributo del contexto tiene un cierto valor, se debe modificar una parte específica del proceso.

A continuación, se describe detalladamente cómo funciona este método, que es representado por la figura 4:

1. **Ajuste del Proceso Inicial al Contexto del Proyecto:** El primer paso en el método DynaTail es el ajuste del proceso de desarrollo estándar al contexto específico del proyecto. Esto implica analizar el modelo de contexto del proyecto, que contiene las variables clave que influyen en el desarrollo. Basándose en este análisis realizado con las reglas de tailoring, el proceso estándar se modifica para alinearse mejor con las características únicas y los requisitos del proyecto.
2. **Evaluación del Proceso Ajustado:** Una vez que el proceso se ha ajustado al contexto del proyecto, el siguiente paso es evaluar este proceso ajustado en relación con un objetivo específico. Esto implica utilizar el grafo de influencias para evaluar si el proceso modificado tiene un valor aceptable para el atributo a ser optimizado.
3. **Ajuste Continuo del Proceso o del Contexto:** En función de los resultados de la evaluación, DynaTail permite realizar ajustes adicionales tanto en el proceso como en el contexto del proyecto. Este es un proceso iterativo, donde se realizan cambios en las actividades del proceso, los roles involucrados, o incluso en los aspectos del contexto del proyecto, como los recursos disponibles o la tecnología utilizada. Estos ajustes se realizan con el objetivo de optimizar aún más el proceso para lograr los resultados deseados.

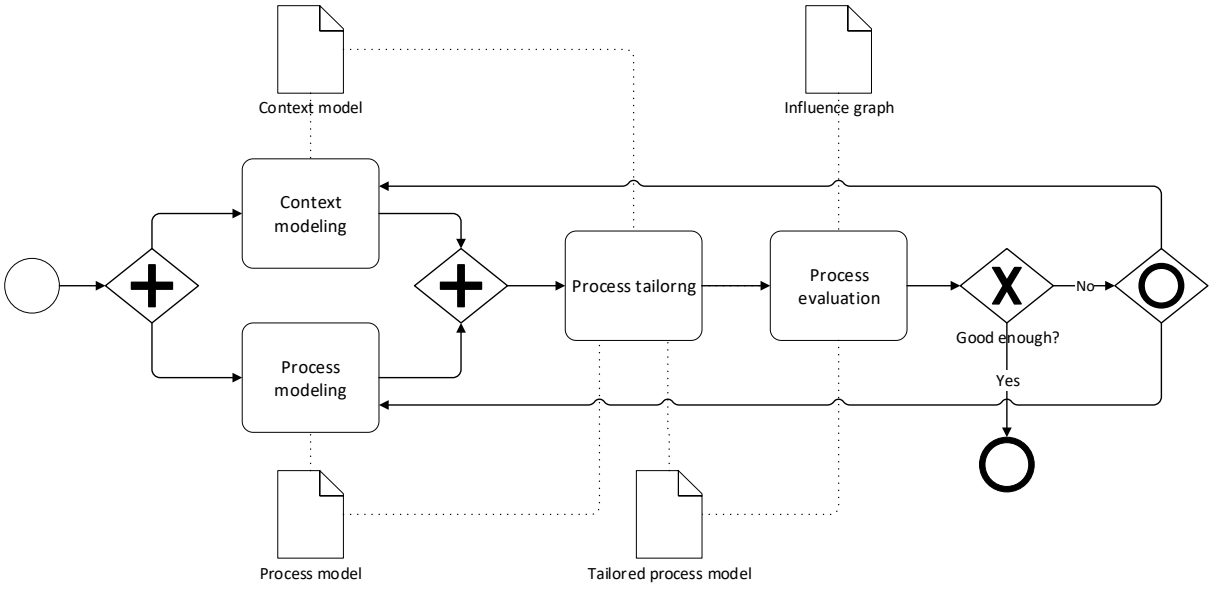


Figura 4: Método de Adaptación Dinámica para Procesos Híbridos - DynaTail [39]

La clave del método DynaTail radica en su capacidad para iterar y refinar continuamente el proceso de desarrollo de software, asegurando que esté en sintonía con las necesidades y restricciones específicas del proyecto. Esto se logra a través de un enfoque sistemático y flexible que permite una adaptación ágil y precisa del proceso.

En resumen, DynaTail proporciona un marco robusto para la personalización de los procesos de desarrollo de software, asegurando que sean lo más eficientes y efectivos posible en el contexto de un proyecto específico.

Además, DynaTail se diferencia de soluciones anteriores en que busca unificar las distintas partes para el tailoring de procesos, con un enfoque práctico y moderno, haciendo uso del estándar BPMN 2.0 para lograr esto. Esto contrasta con soluciones existentes que usaban SPEM[16], un lenguaje estándar de OMG (Object Management Group) para el modelamiento de procesos de software, ya que actualmente, no cuenta con herramientas mantenidas que permitan modelar estos procesos.

Sin embargo, la implementación de DynaTail sigue estando en construcción, con partes siendo activamente desarrolladas, como por ejemplo un módulo capaz de generar automáticamente las transformaciones necesarias para el tailoring del proceso en base al contexto específico del proyecto.

3. Descripción de la solución

3.1. Requisitos

Para comenzar la explicación de la solución, primero se deben definir de forma clara cuáles son los requisitos que esta herramienta busca cumplir.

Así, de forma general, se espera que la implementación de la herramienta sea capaz de generar modelos de contexto de proyectos, que a su vez posteriormente puedan ser utilizados como input para realizar el tailoring de un proceso. Además, la interfaz gráfica de la implementación debe ser capaz de conducir todo el flujo del método propuesto, dando así una base concreta sobre la cuál se puedan llamar los demás módulos de DynaTail.

3.1.1. Requisitos específicos

A continuación se listan los requisitos necesarios para lograr los objetivos de este trabajo:

1. Desarrollar una aplicación capaz de definir y generar modelos de contexto de un proyecto. Para esto, debe ser capaz de:
   - Agregar dimensiones al modelo de contexto.
   - Agregarle y eliminarle atributos a una dimensión.
   - Añadirle un número arbitrario de opciones a un atributo, definiendo así el modelo de contexto de la organización.
   - Permitir la selección de una opción por atributo.
   - Instanciar el modelo de contexto del proyecto a partir de las opciones seleccionadas, generando un archivo XMI con la representación del modelo.
2. Desarrollar una aplicación capaz de recibir y manejar un proceso en formato BPMN. Para esto, debe ser capaz de:
   - Recibir un archivo en formato BPMN.
   - Visualizar el proceso a partir del archivo.
3. Integrar el inyector desarrollado por Andrés Wallberg [40] a la aplicación, cuya descripción detallada y funcionalidad están descritas en la sección 3.2.5, dándole de input el proceso subido en formato BPMN, y guardando el archivo del modelo del proceso en formato XMI, producto del inyector.
4. Visualizar grafos de influencias en la aplicación, a partir de su archivo en formato XMI.
5. Permitir seleccionar la característica que se busca optimizar dentro de la aplicación, eligiendo además el tipo de optimización.
6. Tener una sección dentro de la aplicación para visualizar el resultado de dicha optimización y evaluación.

3.2. Tecnologías utilizadas

3.2.1. Lenguajes de programación

Se han utilizado los siguientes lenguajes de programación en el desarrollo de este trabajo:

- **HTML y CSS**: HTML es el lenguaje de marcado estándar para la creación de páginas web, ampliamente utilizado en cualquier aplicación web. CSS, por otro lado, se utiliza para controlar el estilo y el diseño visual de las páginas HTML. La elección de HTML y CSS es natural para cualquier proyecto que implique una interfaz de usuario en la web, por lo que fueron utilizados para proporcionar la estructura y el estilo necesarios a la aplicación.
- **Typescript**: Typescript es un lenguaje de programación que extiende JavaScript añadiendo tipado estático. Al tratarse de una aplicación web, también es estándar el uso de JavaScript para la implementación de sus funcionalidades, por lo que desde un comienzo se planteó utilizar este lenguaje como base. Sin embargo, la elección de TypeScript por sobre JavaScript se debió a que mientras que JavaScript es un lenguaje de tipado dinámico, TypeScript asegura que los tipos de variables se definan y verifiquen durante la escritura del código, lo que mejora la legibilidad y reduce errores en tiempo de ejecución. Aunque cualquier código JavaScript es válido en TypeScript, este último introduce características avanzadas como tipos e interfaces, mejorando la experiencia del desarrollador con herramientas como autocompletado de código y refactorización más eficiente, mejorando además la mantenibilidad y calidad del código.

3.2.2. Frameworks

Para el desarrollo de la aplicación, se optó por utilizar frameworks debido a que estos proporcionan estructuras y herramientas preestablecidas, facilitando un desarrollo más eficiente y organizado. A continuación se detalla la elección de los frameworks SvelteKit y Tailwind CSS para este trabajo.

SvelteKit SvelteKit es un framework de aplicaciones web que facilita la construcción de aplicaciones web interactivas y dinámicas. Este framework está construído sobre Svelte, a su vez un framework que hace uso de JavaScript (o TypeScript), HTML y CSS.

La elección de SvelteKit para este proyecto se basa en su arquitectura altamente modular y su sistema de enrutamiento integrado, que simplifica la estructura del proyecto y la gestión de páginas. Además, a diferencia de frameworks como React o Angular, SvelteKit está optimizado para el Server-Side Rendering (SSR), mejorando significativamente el tiempo de carga de la página y permitiendo implementar funcionalidades server-side de forma más sencilla y eficiente, una ventaja clave para el rendimiento general del proyecto.

Tailwind CSS Tailwind CSS es un framework CSS con un enfoque altamente personalizable. A diferencia de otros frameworks CSS, Tailwind CSS proporciona clases de utilidad que se pueden componer para construir diseños personalizados sin salir del HTML.

La elección de Tailwind CSS para el estilo de la aplicación se debe a su capacidad para acelerar el desarrollo de interfaces personalizadas, manteniendo un alto nivel de calidad en el diseño, ya que a diferencia de frameworks más tradicionales, Tailwind CSS evita la necesidad de sobrescribir estilos preexistentes, facilitando un desarrollo más ágil y una mayor coherencia en el diseño a lo largo del proyecto.

3.2.3. Arquitectura del software

La integración de Tailwind CSS con SvelteKit facilita el desarrollo de interfaces de usuario con un diseño consistente y personalizable. Tailwind, al proporcionar clases de utilidad a bajo nivel, permite un control detallado del diseño directamente en los archivos de Svelte, lo que resulta en una experiencia de desarrollo dónde las distintas partes de una página se pueden implementar en un mismo lugar.

A su vez, TypeScript juega un papel crucial en la arquitectura del software. Su uso en el proyecto garantiza la legibilidad y mantenibilidad del código, aspectos esenciales considerando que otros desarrolladores trabajarán sobre este código en el futuro. El tipado estático de TypeScript ayuda a prevenir errores comunes en tiempo de compilación y proporciona una documentación clara del código, facilitando la integración y expansión de los módulos de DynaTail a medida que estos se desarrollen.

SvelteKit, siendo el núcleo del desarrollo, promueve una estructura modular en la construcción de la aplicación. Esta modularización no solo fomenta la reusabilidad de componentes, sino que también simplifica la escalabilidad y el mantenimiento de la aplicación. La capacidad de SvelteKit para separar la lógica de la aplicación en componentes reutilizables permite hacer uso de buenas prácticas de la programación, como DRY (Don't Repeat Yourself) [15], lo que es vital para la mantención y expansión de la aplicación, especialmente cuando se integren nuevos módulos de DynaTail.

En cuanto a la gestión de datos, se decidió descartar la necesidad de una base de datos tradicional debido a la búsqueda de una solución más simplificada y directa para el manejo de la información.

En conjunto, la arquitectura del software está diseñada para ser robusta, escalable y fácilmente extensible, asegurando que la base del proyecto sirva como un punto de partida sólido para futuros desarrollos y mejoras.

3.2.4. Librerías utilizadas

En el desarrollo de este proyecto, se han incorporado diversas librerías externas, seleccionadas por su funcionalidad específica. A continuación, se presenta una descripción de estas librerías y su papel en la aplicación:

- **bpmn-js:**[5] Utilizada para la visualización y modelado de procesos de negocio mediante el estándar BPMN (Business Process Model and Notation). Su integración fue clave para permitir la visualización interactiva de los procesos en formato BPMN dentro de la aplicación.
- **D3.js:**[28] D3.js es una librería generar visualizaciones complejas e interactivas de datos en la web. Su integración tuvo como objetivo la visualización del grafo de influencias a partir de su XMI.
- **felte:**[3] Esta librería se utilizó para la gestión y validación de algunos formularios, ofreciendo una solución sencilla y extensible para manejar ciertos formularios en la aplicación, tales como los formularios para agregar dimensiones y atributos al modelo de contexto.

- **lodash:**[33] Una biblioteca de utilidades JavaScript que proporciona funciones modulares para tareas comunes de programación. Lodash se utilizó para mejorar la legibilidad del código y optimizar operaciones como manipulación de arrays y objetos.
- **uuid:**[25] Genera identificadores únicos universales (UUID) para uso en la aplicación. Su integración fue necesaria para la generación de los modelos de contexto, ya que cada elemento del modelo generado requería un identificar único.
- **xml2js:**[20] Facilita la conversión de datos XML a formato JavaScript y viceversa. Se utilizó en la aplicación para parsear la información del grafo de influencias en formato XMI.

Además de estas librerías, se ha incluido **Prettier**[34] como herramienta de formateo de código. Prettier ayuda a mantener un estilo de código coherente y legible, lo que es fundamental para la mantenibilidad a largo plazo del proyecto.

3.2.5. Componentes externos

Como uno de los componentes ya desarrollados de DynaTail, el *inyector* desarrollado por Andrés Wallberg [40] fue integrado a la aplicación. Este inyector, implementado en Java y distribuido en formato de archivo JAR (Java ARchive), juega un papel fundamental en el manejo y la transformación de procesos modelados en BPMN.

El inyector tiene la responsabilidad de transformar archivos de proceso BPMN (archivos BPMN, como en la Figura 23) en modelos de proceso BPMN que conforman con el metamodelo BPMN 2.0 (archivos XMI, como en la Figura 24). Este proceso de transformación es vital para asegurar que los modelos de procesos puedan ser manipulados y gestionados adecuadamente dentro de la aplicación.

3.3. Diseño del archivo generado para el contexto

El diseño de los archivos generados para el contexto en este proyecto sigue los estándares utilizados para la representación de un modelo de contexto de un proyecto, utilizando el formato XMI.

La jerarquía de los archivos XMI es la siguiente:

- **Context**: Elemento raíz que encapsula todo el contexto.
- **myDimensions**: Dentro de 'Context', representa las diferentes dimensiones del contexto.
- **myContextAttributes**: Anidado dentro de 'myDimensions', define los atributos del contexto.
- **posibleValues**: Anidado dentro de 'myContextAttributes', enumera los posibles valores de un atributo.
- **myContextConfigurations**: Representa la configuración concreta del contexto.
- **contextAttributeConfiguration**: Anidado dentro de 'myContextConfigurations', detalla la configuración específica de cada atributo, incluyendo el identificador único y el valor asociado al 'posibleValues' seleccionado del atributo.

Cada uno de estos elementos tiene sus propios atributos, como 'name' y 'description'. Para 'posibleValue', además de estos atributos, se incluye 'value'. Los valores de estos atributos se asignan según la configuración realizada por el usuario en el módulo generador de modelos de contexto de la aplicación.

Además, cada elemento tiene un 'id', que es un atributo que sirve como identificador único dentro del contexto.

En los anexos de este informe, se incluye una imágen (Figura 21) que representa un archivo XMI. Esta figura ilustra un modelo de contexto de un proyecto, generado mediante la aplicación desarrollada en este trabajo de título.

3.4. Interfaces

En el desarrollo de este proyecto, se aprovecharon las capacidades de tipado de TypeScript para crear interfaces que modelan estructuras de datos clave: el grafo de influencias y el contexto. Estas interfaces proporcionan una estructura clara y definida, facilitando la manipulación de estos datos, mejorando la legibilidad y también la mantenibilidad del código. En los anexos se incluye una imágen (Figura 22) de cómo se implementaron estas interfaces.

3.4.1. Modelado del Grafo de Influencias

La interfaz `Node` representa un nodo individual en el grafo de influencias, definiendo sus atributos 'type' y 'name'.

Por otro lado, la interfaz `Influence` describe una relación de influencia entre nodos, especificando el 'source' (fuente), el 'target' (objetivo) y el 'value' (valor) de la influencia. Esta interfaz representa cómo los distintos nodos del grafo se relacionan entre sí.

La interfaz `InfluenceGraph` combina estas dos interfaces anteriores, agrupando 'nodes' y 'influences' en una estructura que representa el grafo de influencias completo. Esta interfaz permite manejar el grafo como una entidad unificada, facilitando su visualización la librería D3.

3.4.2. Modelado del Contexto

En cuanto al modelado del contexto, se definieron varias interfaces para representar su estructura y componentes. La interfaz `Context` encapsula la noción del contexto en su totalidad, conteniendo un arreglo de 'dimensions', cada una representada por la interfaz `Dimension`.

La interfaz `Dimension` define una dimensión específica del contexto, incluyendo su 'name' y un conjunto de 'attributes'. Esta interfaz es esencial para categorizar y organizar los diferentes aspectos del contexto.

Por último, la interfaz `Attribute` se utiliza para detallar cada atributo dentro de una dimensión, especificando su 'name' y las 'options' disponibles para ese atributo. Esto permite una representación detallada y flexible de los atributos que componen cada dimensión del contexto.

En conjunto, estas interfaces proporcionan una estructura clara y eficiente para el manejo de estos datos, aprovechando plenamente las capacidades de TypeScript para mejorar la calidad del código y facilitar su mantenimiento y expansión.

4. Implementación de la solución

En esta sección se describe el funcionamiento paso a paso de la aplicación desarrollada, denominada DynaTool, una implementación de DynaTail. Es importante notar que su desarrollo se realizó en inglés, dado que, al utilizar el inglés, la herramienta DynaTool se vuelve accesible a una audiencia más amplia, ya que su uso es estándar en la industria.

4.1. Inicio

La pantalla de inicio (Figura 5) es la primera interacción del usuario con la herramienta DynaTool.

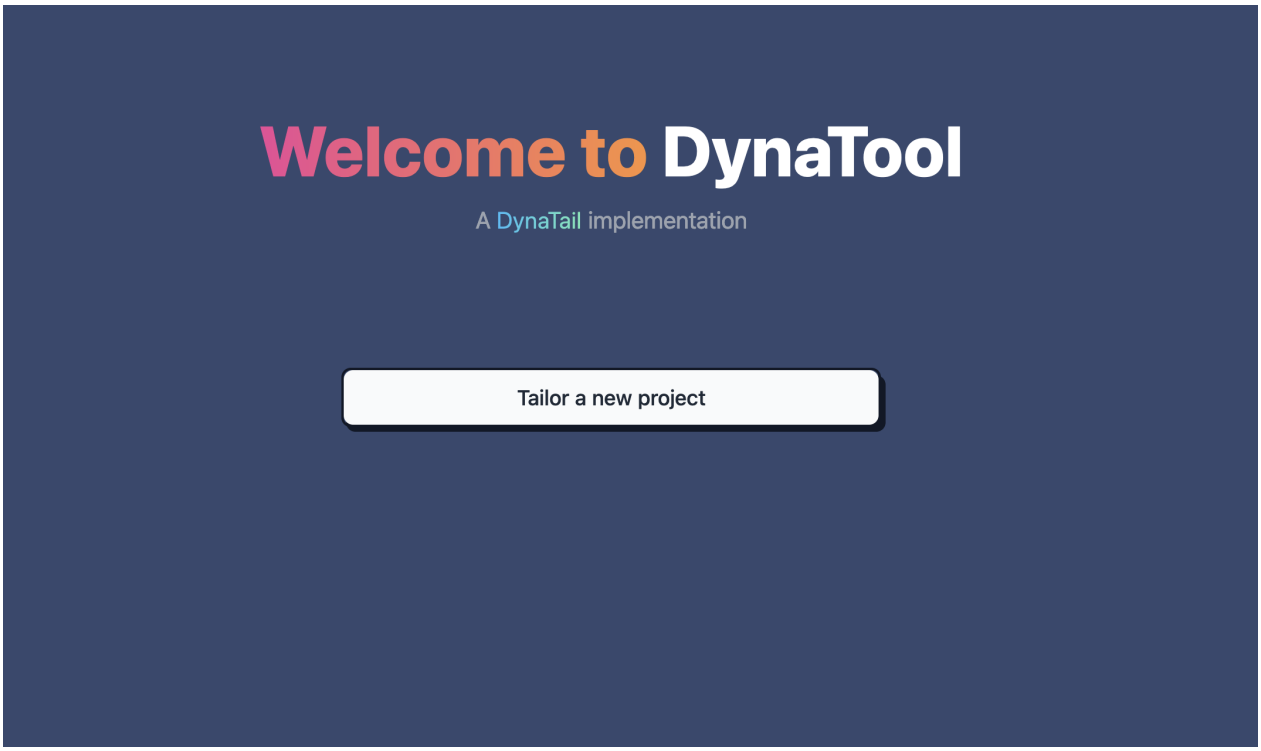
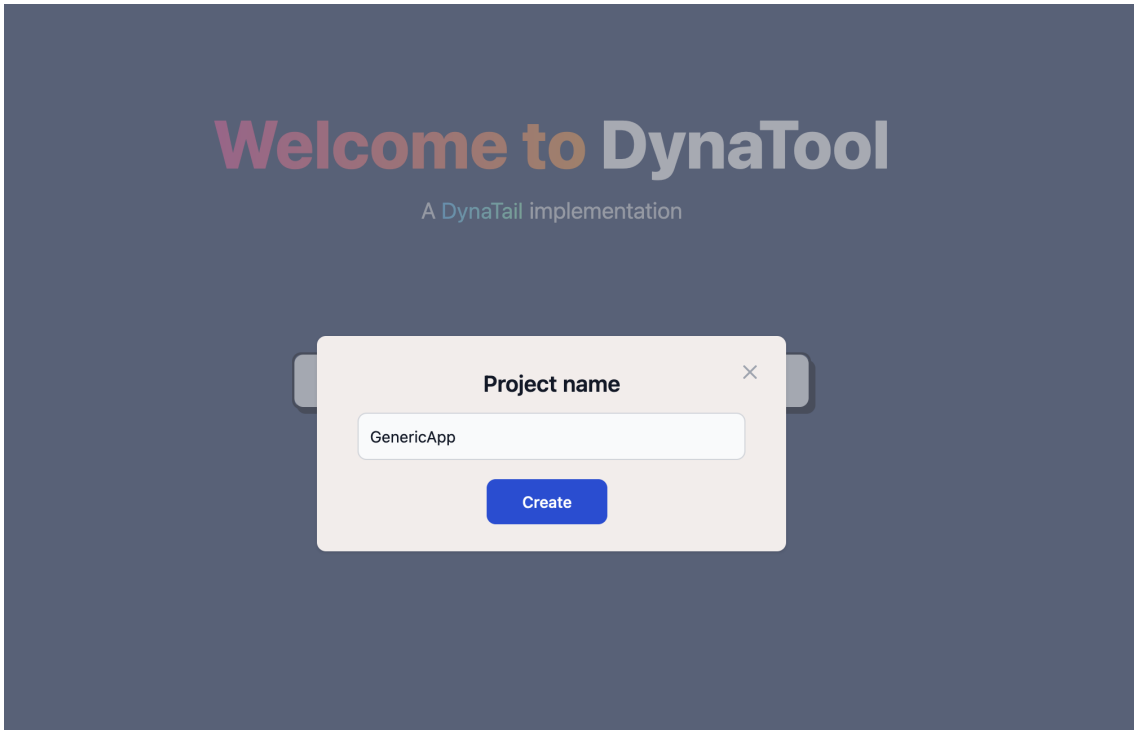
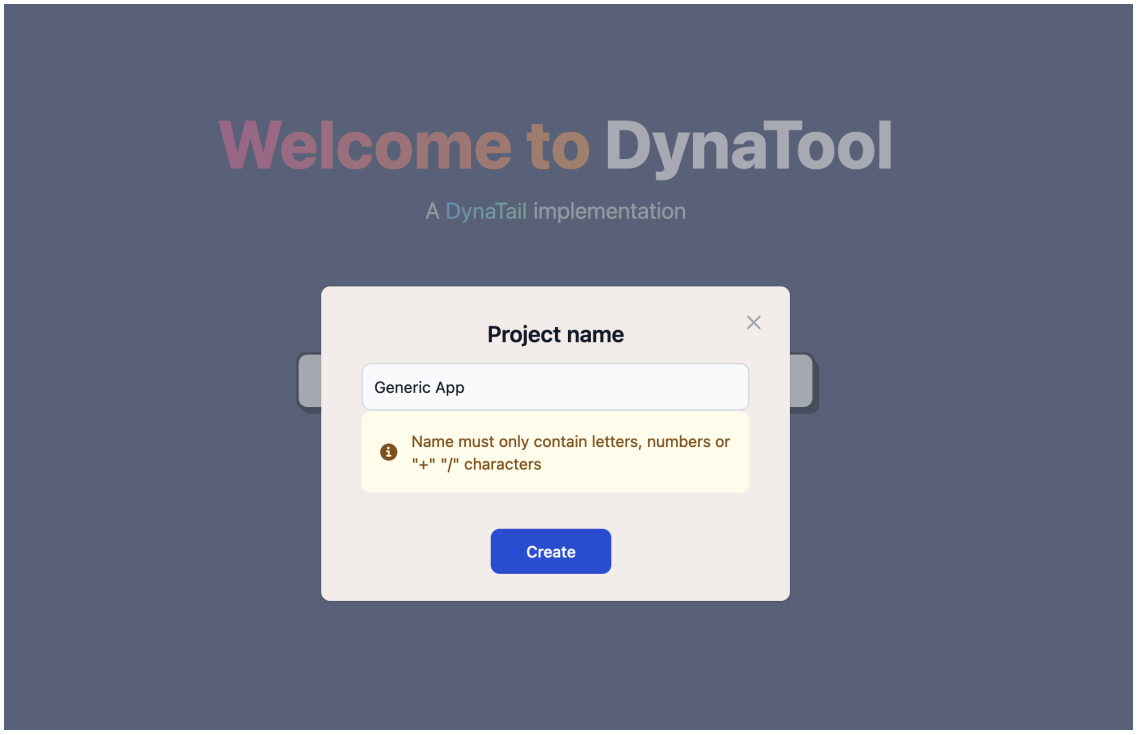


Figura 5: Pantalla de inicio de DynaTool.

En la esta página de bienvenida, se presentan al usuario con la opción 'Tailor a new project' para comenzar el *tailoring* de un nuevo proyecto. Esta interfaz es limpia y minimalista, sirviendo como presentación de la herramienta.



(a) Formulario para ingresar el nombre del proyecto



(b) Advertencia de validación del nombre de un proyecto que contiene espacios blancos

Figura 6: Creación de un nuevo proyecto en DynaTool.

Al seleccionar la opción de comenzar un nuevo proyecto, se presenta al usuario un modal como se muestra en la Figura 6a, donde se solicita el nombre del nuevo proyecto. Una vez que el usuario ingresa el nombre del proyecto, la herramienta realiza un proceso de codificación

a base64 para guardar el nombre.

Esto debido a que la herramienta aprovecha el enrutamiento dinámico proporcionado por SvelteKit para manejar de forma eficiente las URL de los proyectos. SvelteKit permite definir rutas con parámetros variables, conocidos como 'slugs', que se pueden utilizar para cargar y presentar contenido específico basado en la URL de acceso. En el caso de DynaTool, este enrutamiento dinámico se utiliza para incorporar el nombre del proyecto codificado en base64 en la URL, lo que permite una navegación intuitiva y un acceso directo a cada proyecto individual. Esta codificación se emplea por varias razones importantes. En primer lugar, permite manejar de manera segura una amplia gama de caracteres que el usuario puede incluir en el nombre del proyecto, evitando así problemas de codificación en la URL. Al convertir el nombre a base64, se garantiza que el identificador del proyecto en la URL sea compatible con los estándares web, lo que resulta en una URL limpia y con formato estándar, como se muestra en el siguiente ejemplo: `dynatool.com/[nombre_en_base64]/app`.

Además, este enfoque mejora la organización en el servidor, ya que todos los archivos asociados a un proyecto se almacenan en un directorio específico con la codificación base64 del nombre del proyecto, de la siguiente forma: `files/[nombre_en_base64]/`. Esto simplifica la gestión de archivos y aísla los recursos de cada proyecto, evitando conflictos entre nombres y asegurando una estructura de directorios coherente.

Así, si el usuario intenta ingresar un nombre que no cumple con los criterios de validación, como se ilustra en la Figura 6b., DynaTool proporciona retroalimentación inmediata. Esta advertencia garantiza que el nombre del proyecto cumpla con las restricciones establecidas por la codificación de base64, permitiendo sólo los caracteres manejables por esta codificación.

Estas interfaces conducen al usuario a través del proceso de inicio de forma intuitiva, preparando el escenario para las siguientes etapas de modelado de contexto, manejo del proceso y optimización que serán descritas en las secciones siguientes.

4.2. Modelador de contextos

Una vez ingresado el nombre del proyecto y seleccionado el botón de crear, la aplicación conduce al usuario a la interfaz mostrada en la Figura 7, donde se puede gestionar el modelador de contexto y el proceso. En esta sección se abordará la parte del modelador de contexto de esta página.

El modelador de contexto permite a los usuarios definir y personalizar los factores clave que caracterizan el contexto del proyecto. Para esto, cómo se puede apreciar en la Figura 7, al ingresar a esta página se presenta un contexto inicial con distintas dimensiones, atributos y valores de atributos, que puede ser modificado a elección del usuario. 'Project' y 'Team', se tomaron cómo dimensiones base ya que son las más comunes que influyen dentro de un proyecto. Además, cada una de estas dimensiones presenta dos atributos base que pueden ser eliminados según criterio del usuario.

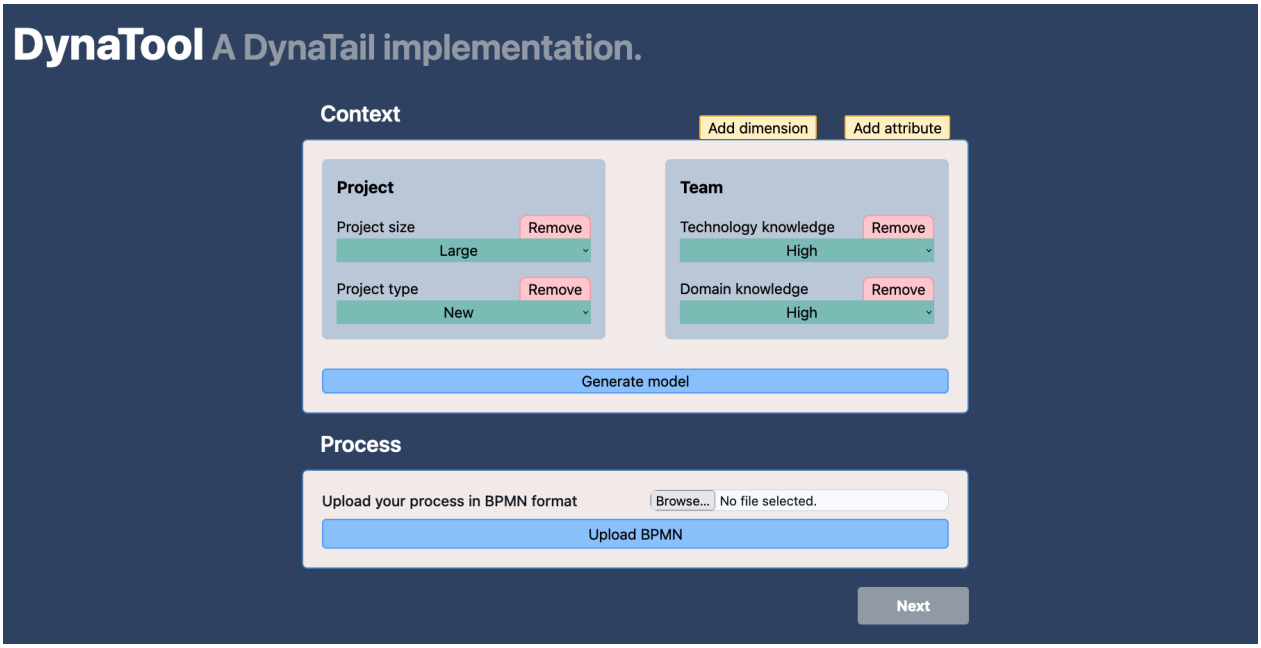


Figura 7: Página de inicio del tailoring del proceso.

Para realizar la modificación de este modelo de contexto, existen tres tipos de botones en esta interfaz. El primero, de color rojo, presenta la posibilidad de eliminar un atributo de una dimensión específica, y se encuentra localizado sobre cada atributo. El segundo, en la parte superior derecha de la interfaz, permite agregarle una nueva dimensión a este contexto. El tercer y último botón, también en la parte superior derecha, permite agregarle un atributo con distintas opciones a una dimensión específica.

En la Figura 8, se muestra la funcionalidad de agregar una dimensión. Esto se hace a través de un modal, que se abre al apretar el botón 'Add dimension', le permite al usuario agregar nuevas dimensiones sobre las cuáles incluir atributos. En esta figura, se le agrega una dimensión 'Product' al contexto.

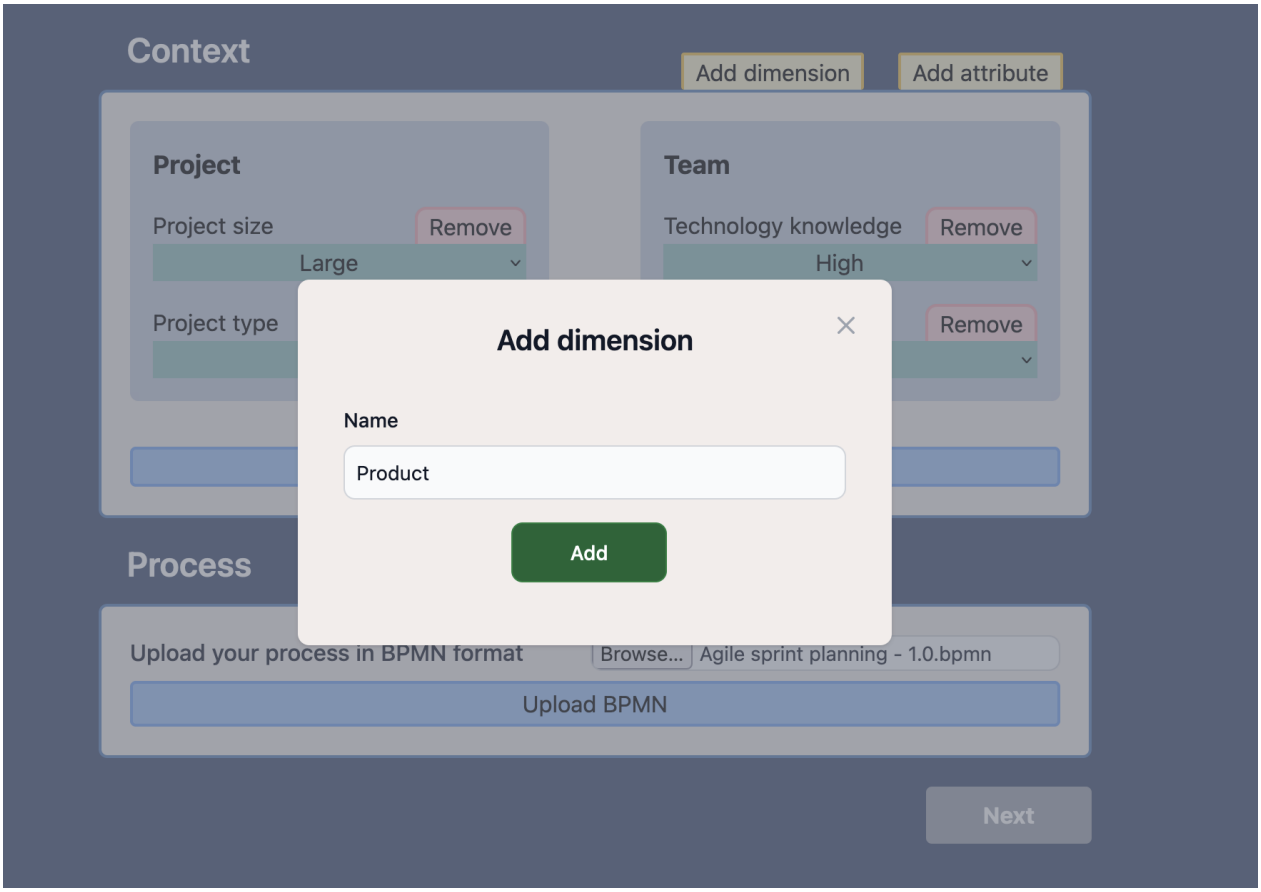


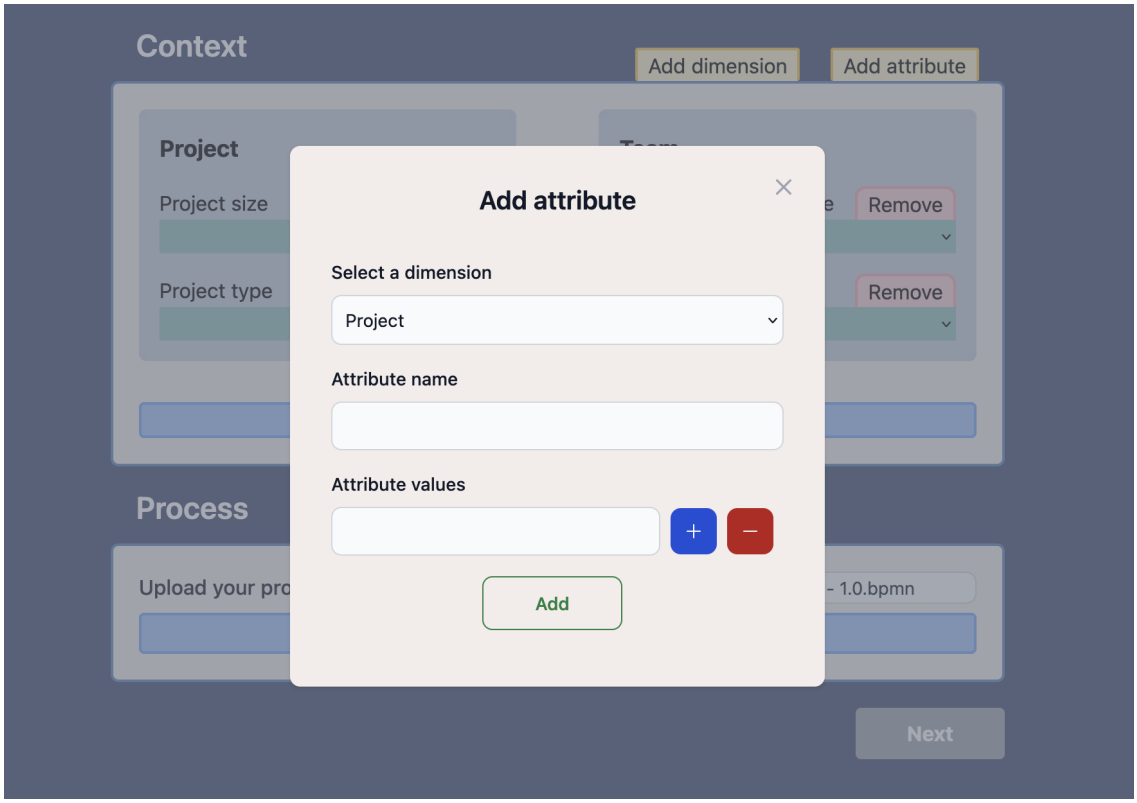
Figura 8: Interfaz para añadir nuevas dimensiones al contexto del proyecto.

Como se ilustra en la Figura 9a., la opción 'Add attribute' abre un diálogo que permite a los usuarios definir nuevos atributos dentro de una dimensión existente. Esto es crucial para refinar el modelo de contexto con especificidades que se ajusten a los parámetros del proyecto actual.

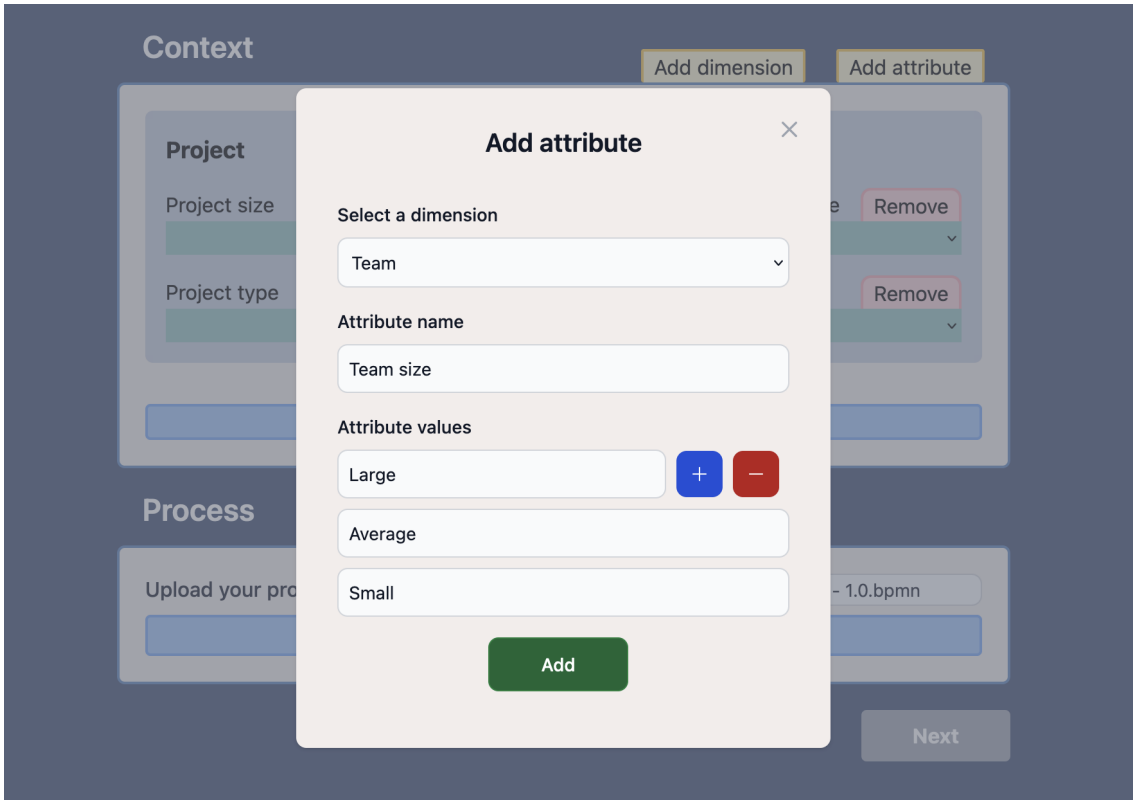
Este formulario tiene tres campos:

1. **Select a dimension:** Permite seleccionar entre las dimensiones creadas a cual se le se desea agregar el nuevo atributo.
2. **Attribute name:** Permite definir el nombre de este atributo.
3. **Attribute values:** Permite definir los posibles valores para este atributo. Este campo además tiene dos botones, uno para permitir agregar un nuevo valor, y otro para disminuir la cantidad de valores posibles.

Es importante notar que dentro del manejo de los posibles valores para el atributo, si se aumenta el número de posibles valores y estos no se llenan, sólo se tomarán en consideración los campos que sí presentan un valor definido. En la Figura 9b., se planea agregar un atributo llamado 'Team size' a la dimensión de nombre 'Team', con posibles valores {Large, Average, Small}. Esta acción se ve reflejada en la interfaz mostrada en la Figura 10, que muestra cómo la interfaz se adapta a medida que aumenta la cantidad de atributos para las dimensiones del contexto.



(a) Diálogo para agregar nuevos atributos al contexto.



(b) Añadir o eliminar valores para un atributo específico.

Figura 9: Interfaz para agregar un nuevo atributo.

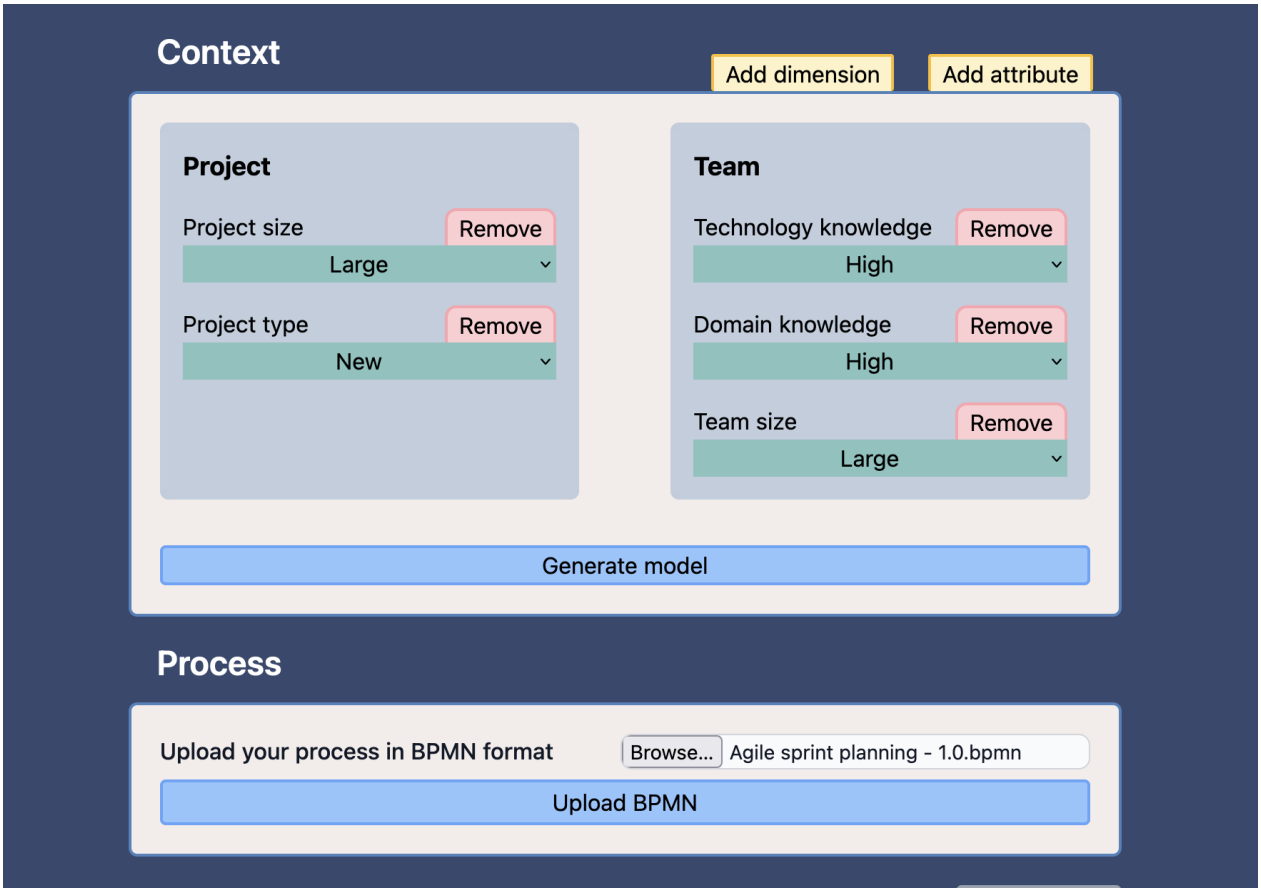
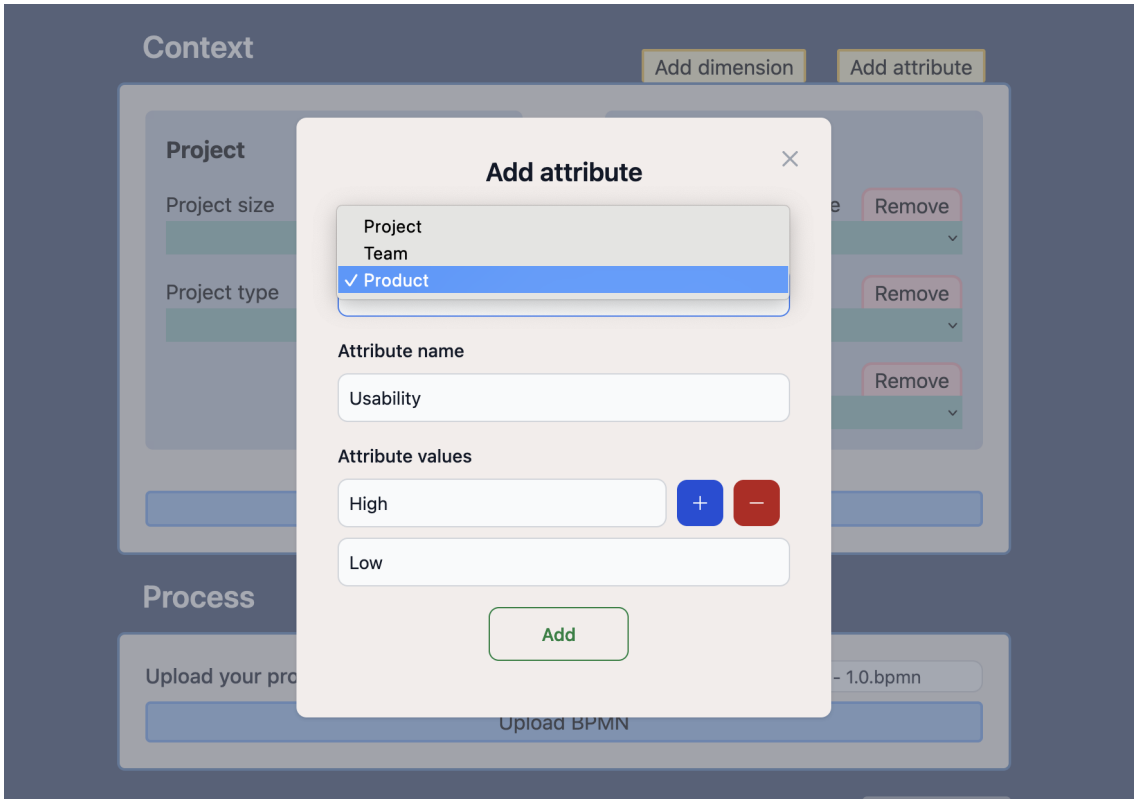
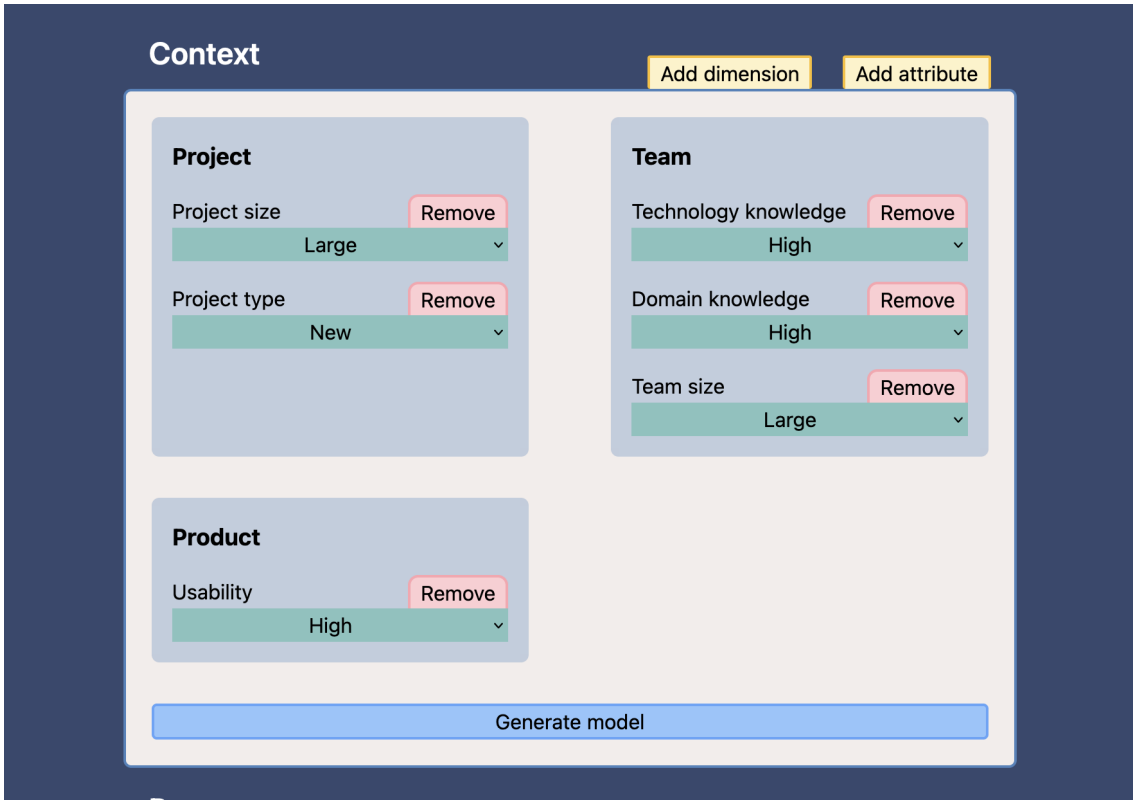


Figura 10: Adición de un nuevo atributo a una dimensión seleccionada.

En la Figura 11a., se muestra cómo funciona la selección de dimensiones, demostrado agregando un nuevo atributo llamado 'Usability' a la dimensión 'Product' agregada previamente (Figura 8), con posibles valores {High, Low}. Este cambio se ve reflejado en la Figura 11b., que muestra cómo se va desarrollando la interfaz a medida que se agregan nuevas dimensiones con atributos.



(a) Gestión de atributos y dimensiones en el contexto del proyecto.



(b) Vista ampliada del modelador de contexto con opciones de configuración.

Figura 11: Interfaz al agregar atributos a nuevas dimensiones

Una vista ampliada del contexto, presentada en la Figura 12, demuestra cómo los usuarios pueden configurar de manera comprensiva el contexto del proyecto. Esto incluye la selección de opciones predefinidas y la personalización a través de la adición de nuevos atributos y dimensiones, lo cual se refleja en una interfaz intuitiva y accesible.

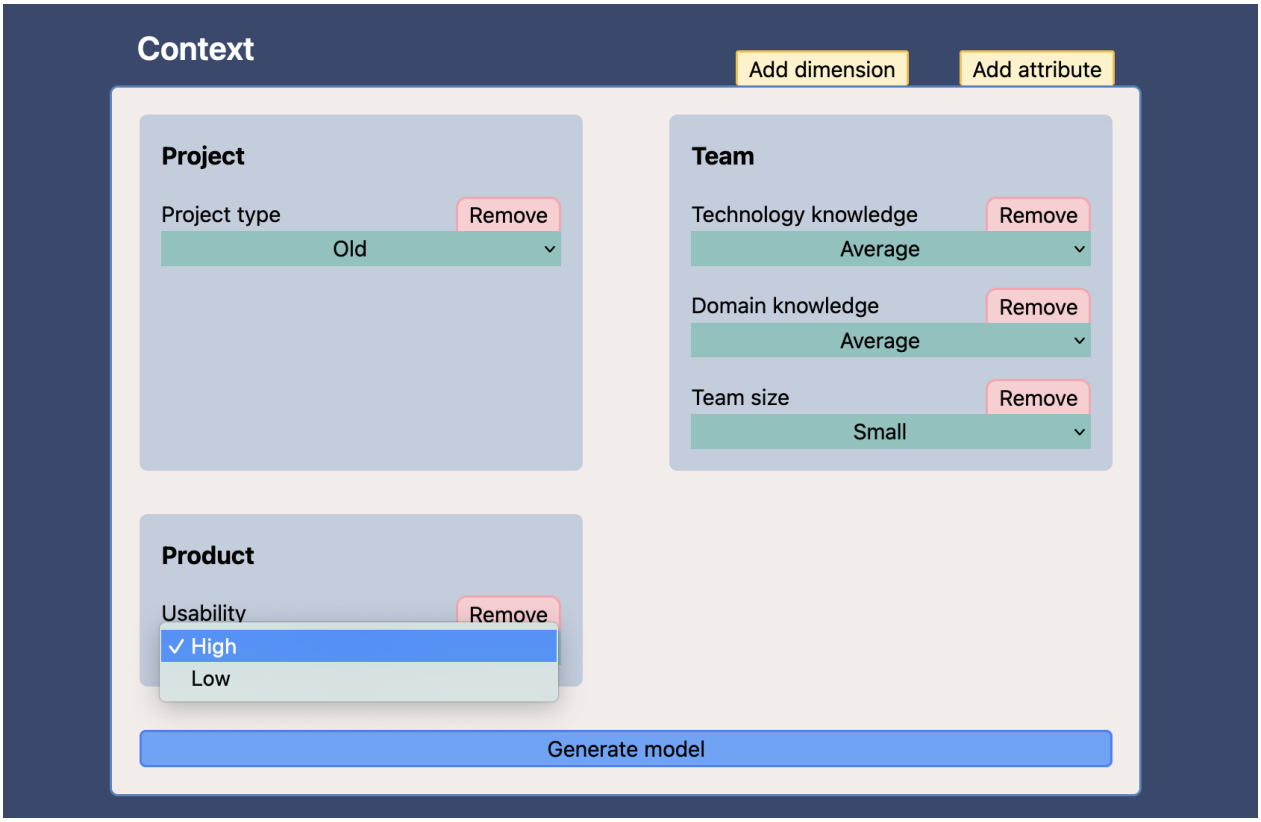


Figura 12: Selección de valores para el contexto de un proyecto específico.

Finalmente, la Figura 12 exhibe cómo los usuarios pueden gestionar las opciones de los atributos, generando así, a partir del contexto de la organización, el contexto del proyecto específico sobre el que se desea trabajar, configurándolo de manera comprensiva, reflejado en una interfaz intuitiva y accesible. Esto se ve a través de la eliminación del atributo 'Project size' de la dimensión 'Project', y la configuración de los demás atributos a distintos valores.

Así, la capacidad de modificar y adaptar de manera dinámica el modelo de contexto del proyecto a través de esta interfaz agiliza el proceso de tailoring. Esto ya que, al apretar el botón de 'Generate model', se genera el modelo de contexto del proyecto en formato XMI necesario para el tailoring, ilustrado en la Figura 21 del anexo A, que presenta la configuración mostrada en la Figura 12, utilizando la estructura definida en la sección 3.3.

Este proceso se logra mediante el uso eficaz de los FormActions de SvelteKit, que permiten enviar los datos recopilados en la interfaz a través de un formulario. La acción asociada al formulario se ejecuta en el lado del servidor, lo que significa que la generación del modelo se realiza de manera segura y eficiente, aprovechando la capacidad de SvelteKit para ejecutar acciones del lado del servidor.

Para la creación del archivo XMI, se recibe la información utilizando la interfaz de contexto definida en la sección 3.4.2, logrando así un manejo fácil de la información contenida en el formulario, dónde sólo es necesario iterar sobre cada campo, asignándoles los elementos correspondientes de la sección 3.3, además de asignarle un identificador único a cada elemento, para luego poder identificarlo dentro de las opciones elegidas. Para esto, se hace uso de un diccionario simple, dónde el valor es el identificador único y la llave es el nombre de la dimensión concatenada al nombre del atributo, permitiendo así que dimensiones distintas puedan tener atributos con nombres iguales sin que estos sean sobre escritos.

Como esta acción se realiza en el servidor, el archivo generado puede ser inmediatamente guardado en el directorio específico del proyecto a través de su codificación en base64.

Para finalizar, vale resaltar que toda esta interfaz se desarrolló de forma modular con distintos componentes, permitiendo así su reutilización posteriormente.

4.3. Manejo del proceso

La interfaz utilizada para la visualización del proceso, mostrada en la Figura 13, permite la selección de un archivo dentro del sistema del usuario. Una vez seleccionado, este se muestra en un campo adyacente y puede ser cargado en la plataforma a través del botón 'Upload BPMN', que a su vez sube y guarda el archivo en el servidor, dentro del directorio específico del proyecto.

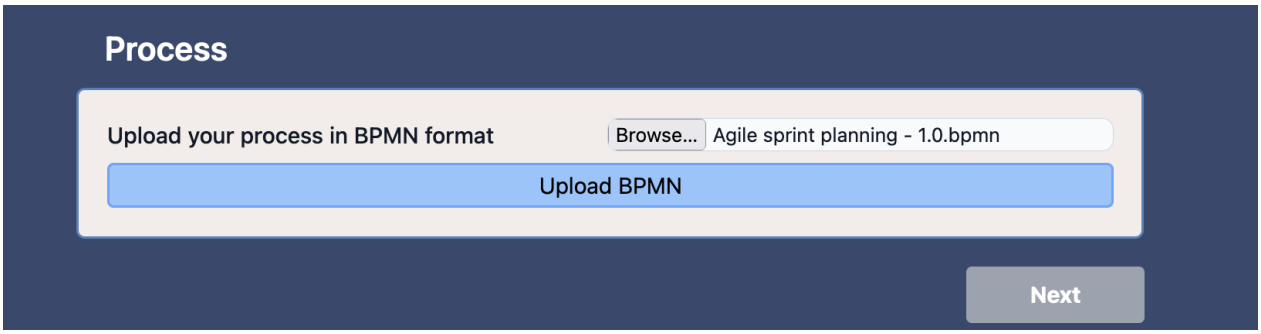
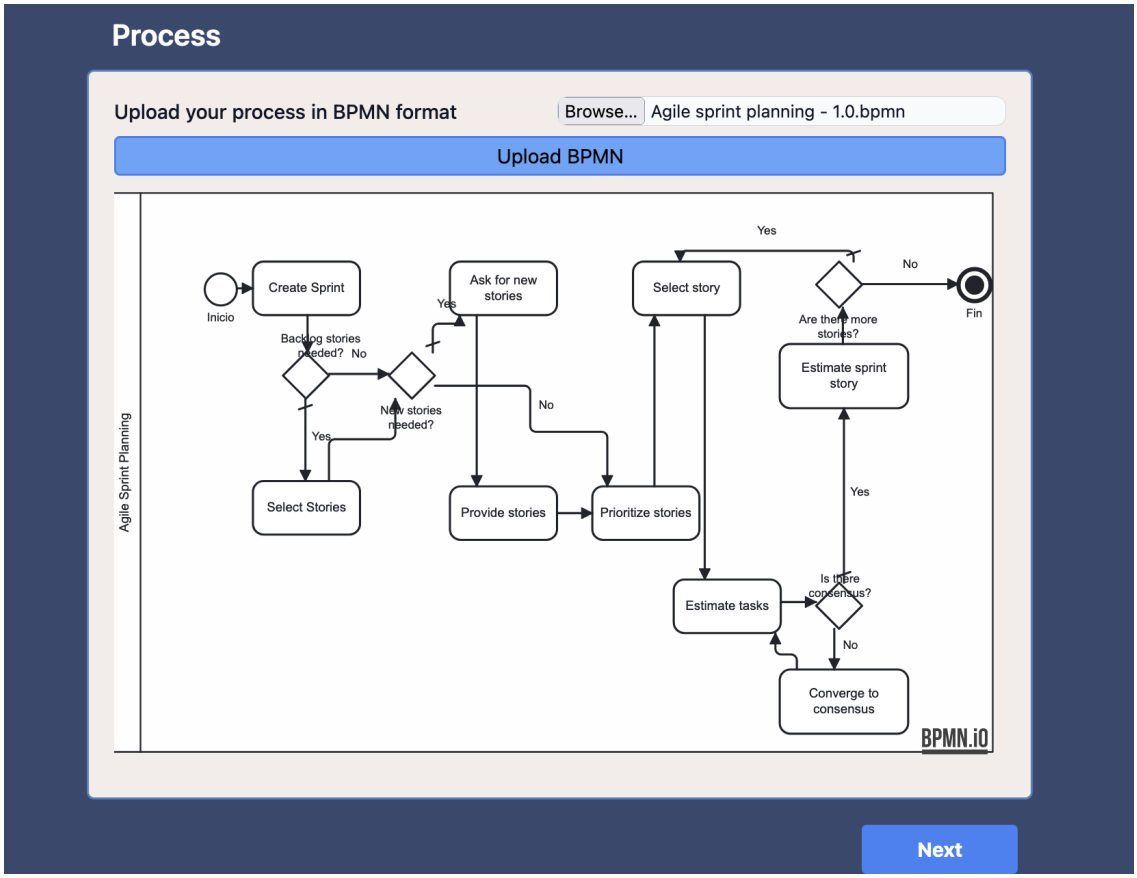
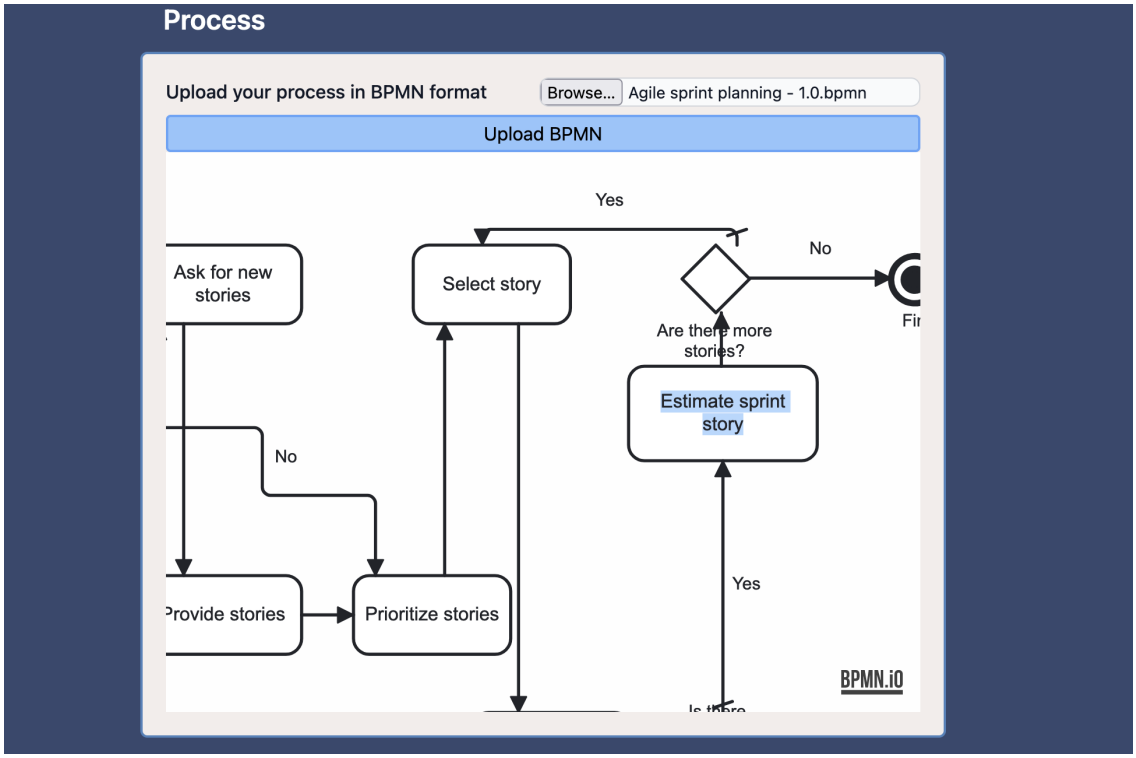


Figura 13: Interfaz de carga del proceso BPMN.

Una vez que el archivo BPMN se carga, la interfaz muestra el diagrama de proceso. Los usuarios pueden visualizar y navegar a través de las diferentes etapas y actividades del proceso de desarrollo, como se ilustra en la Figura 14.



(a) Visualización del proceso BPMN.



(b) Navegación del proceso BPMN.

Figura 14: Visualización y navegación del proceso BPMN.

La herramienta proporciona una representación gráfica que incluye eventos de inicio y fin, tareas, decisiones y flujos de secuencia. Esto permite a los usuarios comprender el flujo del proceso de desarrollo de una manera interactiva, permitiendo la selección de los elementos, la movimentación y el manejo del zoom dentro del proceso.

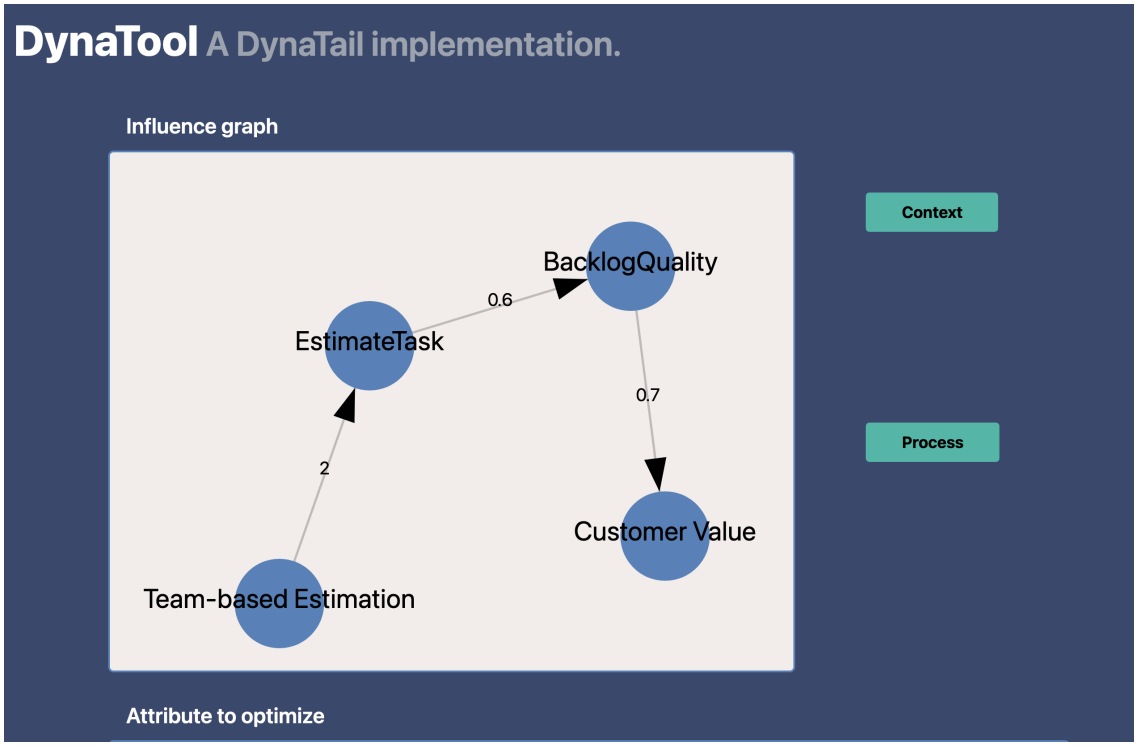
La integración de la librería `bpmn.io` en la interfaz de la herramienta ha demostrado ser una elección robusta y eficaz para la manipulación de modelos BPMN. La capacidad de cargar y visualizar procesos proporciona un valor significativo a los equipos de software en su esfuerzo por realizar el tailoring de estos a través de la herramienta.

Por detrás, la herramienta, además de guardar el archivo, lo utiliza de input para la ejecución del inyector definido en la sección 3.2.5, generando así además el archivo XMI con el modelo del proceso, que posteriormente recibirá el tailoring. La ejecución del inyector también se realiza en el servidor, invocando al JAR que contiene el ejecutable del inyector, guardado entre los archivos de la herramienta. Para el ejemplo de la Figura 14, el archivo subido se encuentra en el anexo C, mientras que el archivo resultante de la ejecución del inyector se encuentra en el anexo D.

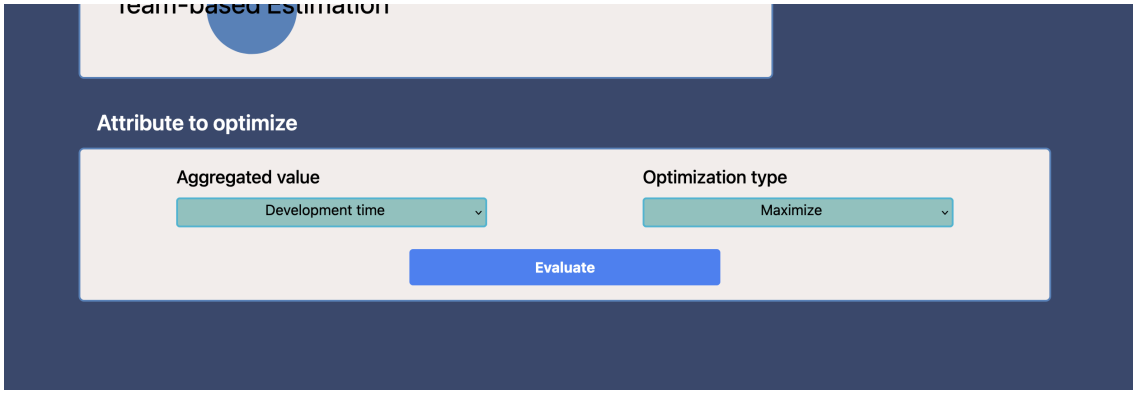
Una vez subido el archivo, el botón 'Next' es habilitado, permitiendo al usuario ir a la siguiente página, dónde se realizará la evaluación.

4.4. Grafo de Influencias

Así, al llegar a esta página, el usuario es presentado con la interfaz mostrada en la Figura 15, donde se puede visualizar un grafo de influencias, gestionar el modelador de contexto y el proceso a través de modales, cómo mostrado en la Figura 15a., y realizar la evaluación del proceso de acuerdo a la métrica seleccionada, cómo mostrado en la Figura 15b.



(a) Vista para visualización de grafo de influencias y botones para abrir modales del contexto y proceso.



(b) Vista para seleccionar el atributo que se busca optimizar.

Figura 15: Página para realizar la evaluación del proceso.

En la Figura 16, se presenta al usuario la interfaz interactiva que muestra el grafo de influencias. Este grafo se ha implementado utilizando la librería D3.js, mencionada en la sección 3.2.4, haciendo uso de la interfaz definida en la sección 3.4.1 para traducir la información contenida en el archivo XMI del grafo a componentes graficables por la librería.

Para la visualización del grafo de influencias, se optó por utilizar un grafo de fuerzas, ya que este ofrece una representación intuitiva y manejable de las relaciones y la ponderación entre los distintos elementos del proceso de desarrollo.

El grafo en sí es completamente interactivo, permitiendo al usuario navegar y explorar las diferentes conexiones y nodos con facilidad. Es posible arrastrar nodos para reposicionarlos,

demostrado al notar la diferencia entre las Figuras 15a. y 16, lo que ayuda a entender mejor las interdependencias y la estructura del grafo. Las aristas que conectan los nodos indican la relación y la dirección de la influencia, y cada arista está etiquetado con un peso numérico que representa la magnitud de esta influencia.

Para este caso específico, el grafo mostrado es un ejemplo simples, ya que aún no ha sido desarrollado el módulo de DynaTail que realizará la evaluación de un proceso, por lo que el uso del grafo en este trabajo de título es una base para futuras implementaciones. Sin embargo, este grafo de ejemplo sirve para ilustrar cómo la herramienta puede utilizarse para visualizar y manipular grafos de influencia más complejos una vez que estén disponibles, ya que la función desarrollada para esta representación es aplicable a cualquier archivo XMI que siga el metamodelo de un grafo de influencias.

El grafo de la Figura 16 es una representación del archivo en el Anexo E, que sirve para ilustrar los elementos necesarios para su visualización como un grafo de fuerzas. Son estos, un elemento **dynatail:CGModel**, que contiene los distintos nodos e influencias. Estos nodos a su vez son elementos llamados **cgnode**, conteniendo el tipo de información que contiene y su nombre, mientras que las influencias son elementos llamados **cginfluence**, que contienen la referencia respecto de cuáles nodos conecta y su peso.

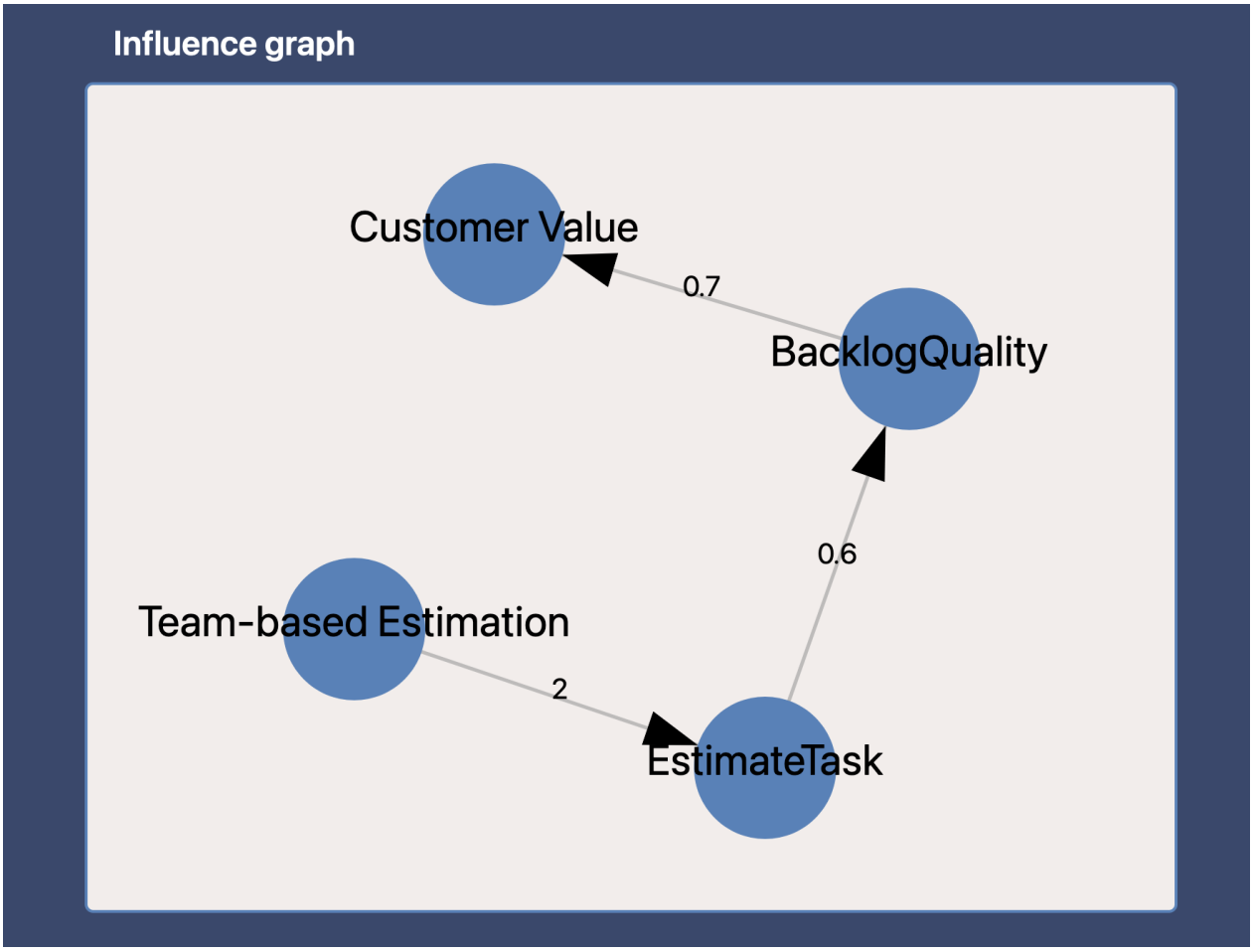


Figura 16: Visualización y navegación del grafo de influencias.

4.5. Optimización

4.5.1. Modal de Contexto

Al apretar el botón 'Context' en la Figura 15, se abre el modal mostrado en la Figura 17. Este tiene como objetivo que el usuario pueda realizar cambios al modelo de contexto dentro de esta misma página.

La implementación de este modal es facilitado por el hecho de que el módulo de creación del modelo de contexto se hizo como un componente reutilizable, por lo que no fue necesario un mayor desarrollo, salvo colocar este componente dentro de otro componente reutilizable: el del modal, utilizado también para todos los modales de la aplicación.

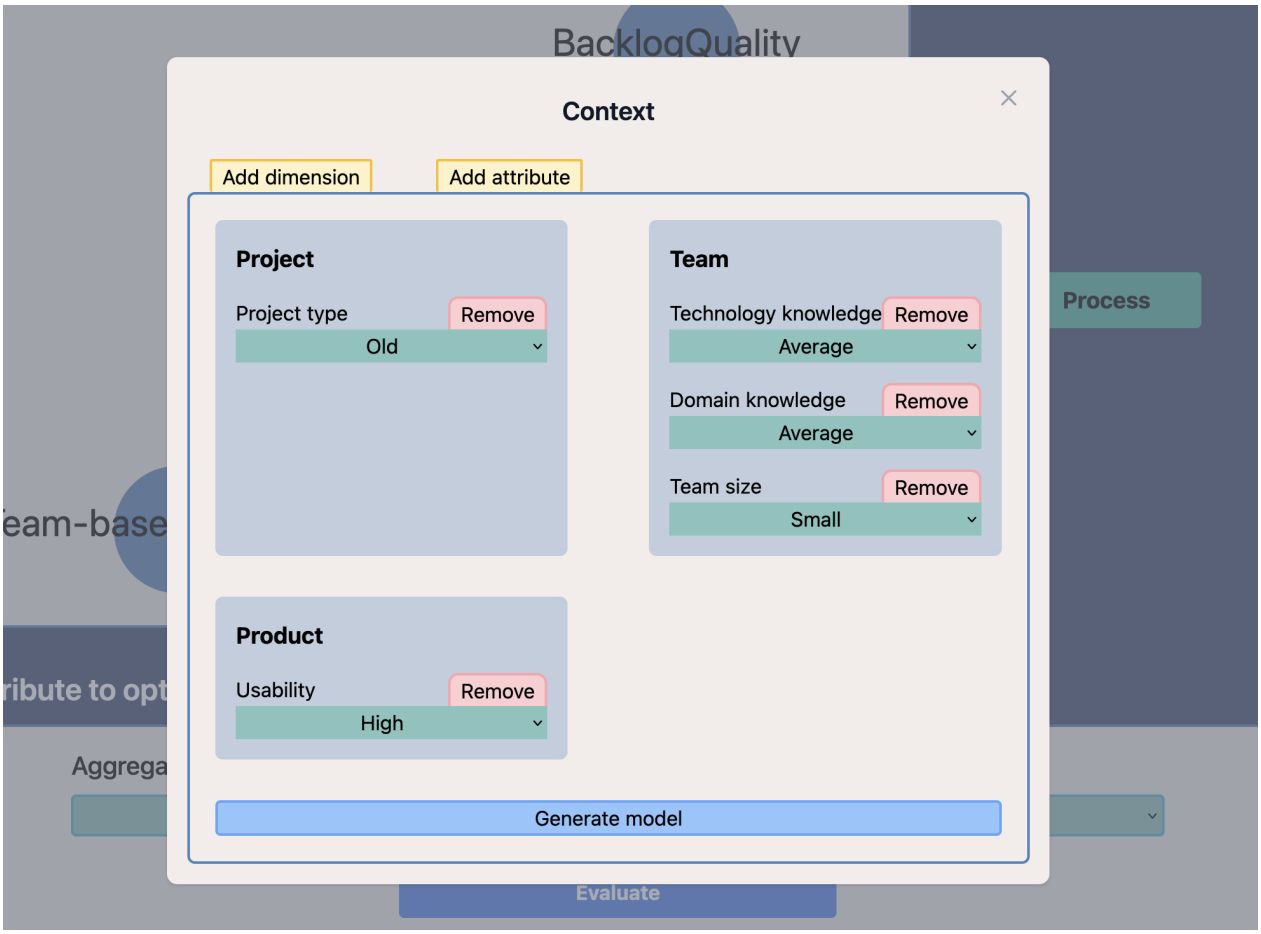


Figura 17: Modal para el modelo del contexto del proyecto.

4.5.2. Modal de Proceso

Análogamente al modal del contexto, al apretar el botón 'Process', se abre el modal mostrado en la Figura 18. Este, de forma similar al anterior, tiene como objetivo que el usuario pueda modificar el proceso subido a la herramienta.

Este desarrollo también fue facilitado por la modularización de los componentes de la aplicación, logrando así un buen uso del principio DRY (Don't repeat yourself)[15].

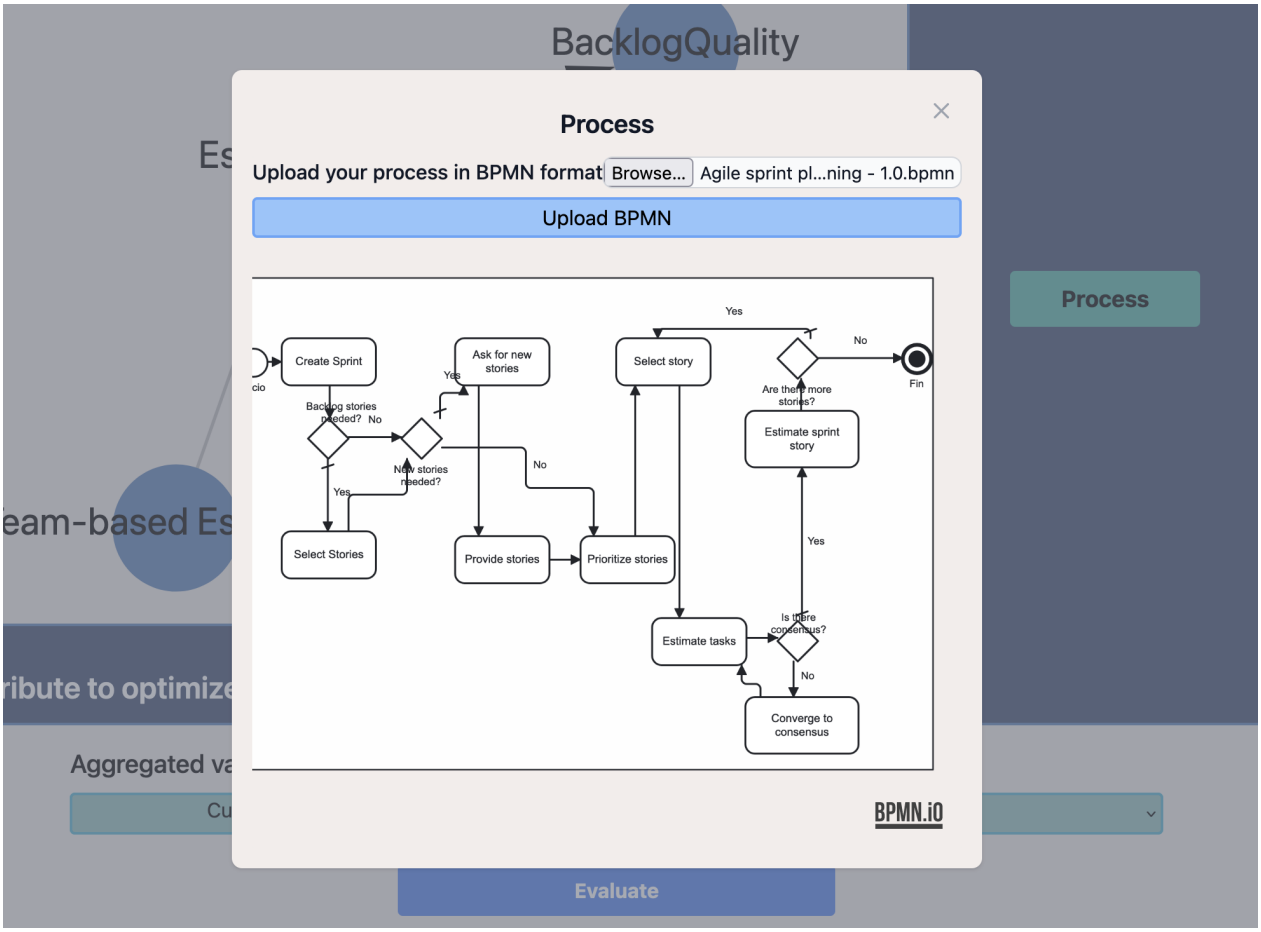


Figura 18: Modal para visualización del proceso.

4.5.3. Evaluación del proceso

En la Figura 19, se introduce al usuario a la fase de la evaluación del proceso dentro de la herramienta DynaTool, donde se puede seleccionar el parámetro que se busca evaluar en la optimización del proceso de desarrollo de software.

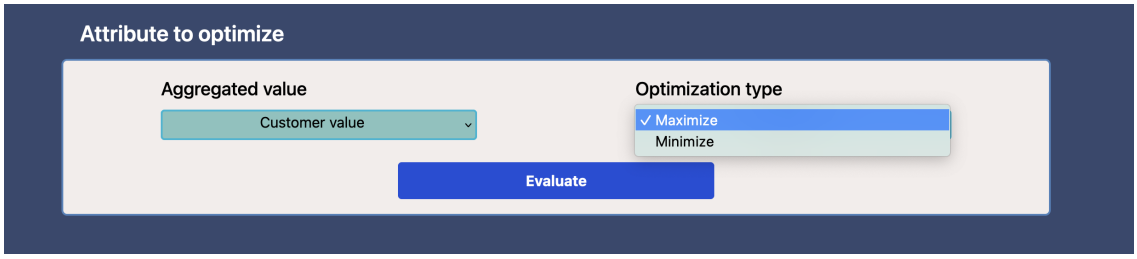
La interfaz gráfica, como se muestra en la Figura 19a., proporciona un mecanismo interactivo donde se puede seleccionar el atributo a optimizar, tales como el valor para el cliente o el tiempo de desarrollo. Así, el usuario puede elegir entre si busca maximizar o minimizar dentro de la evaluación el atributo seleccionado utilizando un menú desplegable.

Una vez que se han establecido estos parámetros, el usuario puede proceder con la evaluación haciendo clic en el botón 'Evaluate'. La Figura 19b. ilustra la interfaz desplegada post-evaluación, que presenta los resultados de la optimización en términos cuantitativos. Por ejemplo, si el atributo seleccionado fue el valor para el cliente y se optó por maximizar, la herramienta calcula y muestra un valor agregado, como 1.17, que representa el resultado de la optimización basado en el grafo de influencia y los datos del contexto proporcionados.

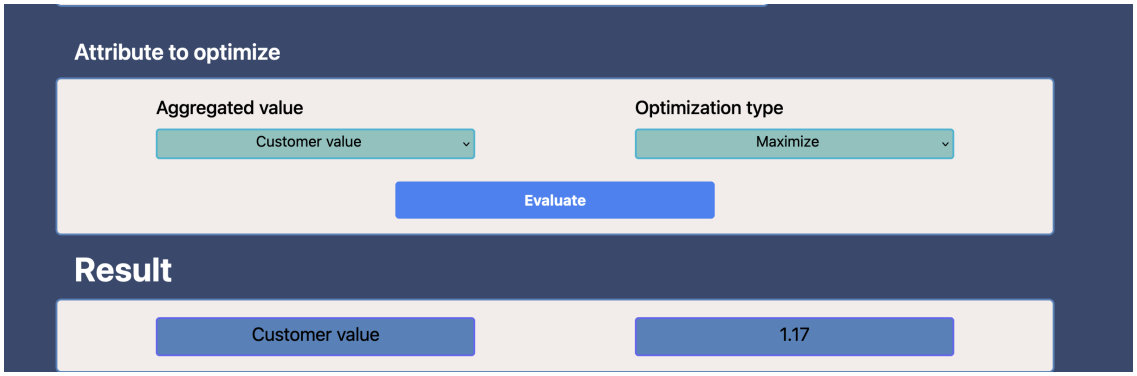
Para realizar esta evaluación, los datos del formulario son enviados a una función implementada en el lado del servidor, que realiza la evaluación del proceso y retorna un número.

Esto ya que el módulo de DynaTail para la evaluación del proceso aún no se encuentra completamente desarrollado, por lo que la interfaz desarrollada sirve cómo un componente sobre la cuál se debe llamar a este módulo una vez se encuentre desarrollado.

Así, si el valor retornado no fuera satisfactorio, dentro de esta misma página el usuario podría modificar los parámetros necesarios, ya sea subiendo un nuevo proceso modificado o cambiando algún atributo del contexto del proyecto, permitiendo realizar una nueva evaluación de este proceso.



(a) Interfaz para seleccionar atributo a optimizar y tipo de optimización.



(b) Vista con resultado generado de realizar la evaluación del proceso.

Figura 19: Interfaz para realizar la optimización y evaluación del proceso.

5. Evaluación

Para evaluar la efectividad de la aplicación desarrollada en resolver el problema planteado, se debe evaluar tanto el modelador de contextos de un proyecto como la efectividad de la interfaz gráfica para articular los distintos módulos de DynaTail.

5.1. Modelador de Contexto

Para realizar la evaluación del módulo desarrollado para la generación de modelos de contexto de proyectos, se hizo uso de la siguiente metodología:

1. Utilizando la aplicación, se crearon varios modelos de contexto de proyectos, variando dimensiones, atributos y posibles valores.
2. Cada uno de estos modelos generados en formato XMI fue ingresado a la herramienta Eclipse Modeling Framework (EMF).
3. Dentro de la herramienta EMF, estándar para la creación y manejo de modelos, se verificó que los datos mostrados para el modelo generado efectivamente fueran los datos seleccionados a la hora de generar el modelo en la aplicación.
4. Esto se hizo verificando que todas las dimensiones, atributos y opciones del contexto de la organización estuvieran presentes en el modelo. Además, para la verificación del modelo del contexto del proyecto, se observó que la configuración especificada para cada atributo fuera efectivamente la opción elegida a la hora de generar el modelo.

Utilizando esta metodología para realizar la validación de la implementación, se lograron resultados positivos con los distintos modelos utilizados, logrando reproducir de manera correcta modelos representativos establecidos previamente.

Como ejemplo concreto de esto, la Figura 20 presenta el modelo de contexto del proyecto generado por la configuración de la Figura 12b.. A su vez, este modelo generado se encuentra en el Anexo A. Así, utilizando la herramienta EMF, se puede verificar que el modelo generado presenta las tres dimensiones definidas en la configuración: {Project, Team, Product}. Además, se verifica que 'Project type' es el único atributo dentro de la dimensión 'Project', y presenta como opciones {New, Old}. También en la Figura 20, se puede verificar que dentro de la dimensión 'Team' se encuentra el atributo Technology knowledge, con posibles valores {High, Average, Low}, y también los atributos {Domain knowledge, Team size}, que no se encuentran expandidos para mostrar sus posibles valores en la figura. También se verifica la existencia de la dimensión 'Product', con su atributo 'Usability' y opciones {High, Low}. Finalmente, en la configuración específica del modelo se verifica la existencia de todos los atributos configurados, además de la correcta configuración del atributo 'Project type', que presenta el valor 'Old'.

Este ejemplo es representativo ya que demuestra a la vez todas las funcionalidades desarrolladas para este módulo:

- Se verifica el poder eliminar atributos con sus opciones, una vez que se eliminó el atributo 'Project size' mostrado en la Figura 7.
- Se verifica el poder agregar atributos con un número arbitrario de opciones, ya que el

modelo presenta el atributo 'Team size' generado en la Figura 9b. y mostrado en la Figura 10.

- Se verifica el poder agregarle dimensiones al contexto, una vez que se muestra la dimensión 'Product', agregada en la Figura 8, y a su vez agregarle atributos a esta nueva dimensión, realizado en la Figura 11.
- Se verifica el poder seleccionar distintas opciones para la configuración, demostrado por la elección de la opción 'Old' para el atributo 'Project type', reflejado en la Figura 12.

Como otro ejemplo de la funcionalidad de la aplicación, se replicó la generación de un modelo previamente establecido y verificado, mostrado en el Anexo F. Este modelo generado se encuentra en el Anexo G, dónde se puede ver que ambos presentan exactamente los mismos elementos. La única diferencia entre ambos archivos son los identificadores únicos generados para cada elemento, una vez que estos son generados de forma aleatoria dentro de la aplicación al momento de la ejecución de la función que genera el archivo.

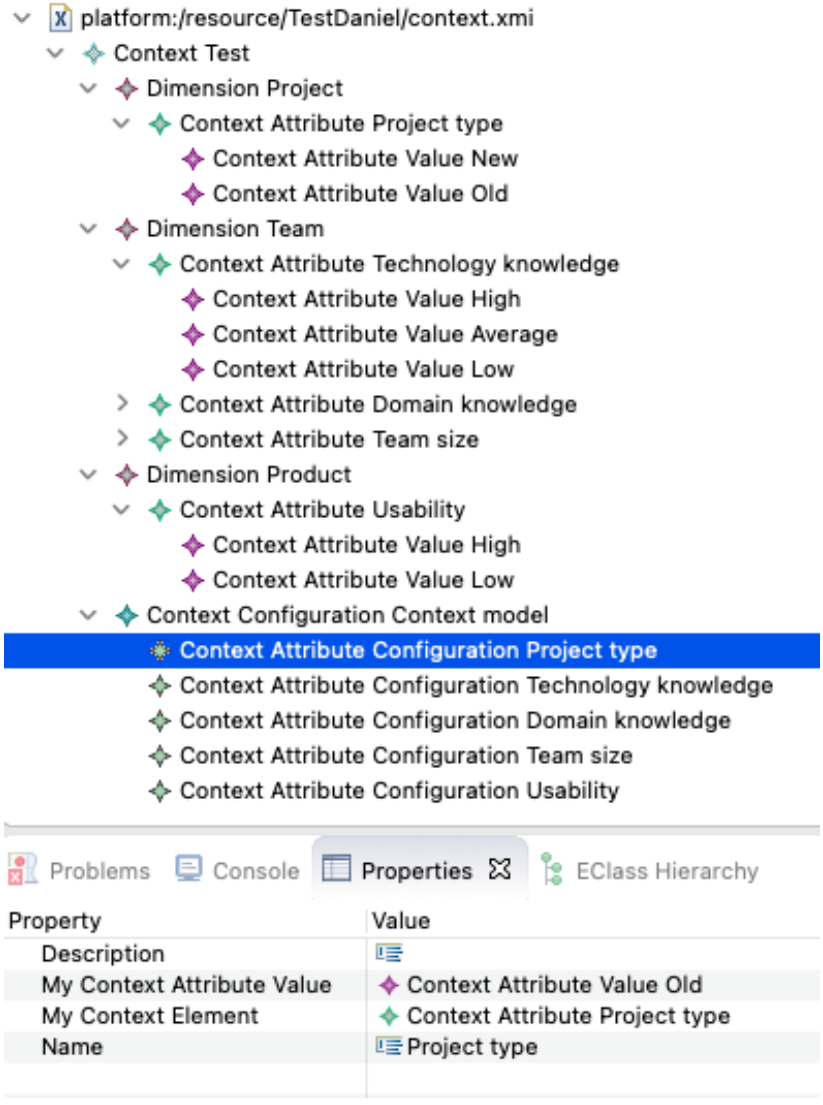


Figura 20: Validación del modelo de contexto mostrado en la figura 21.

5.2. Interfaz Gráfica

La interfaz gráfica de la herramienta de software ha sido diseñada para proporcionar una representación visual e interactiva de las configuraciones resultantes del proceso de tailoring.

5.2.1. Integración y Fluidez en la Interfaz

Cada una de los módulos de DynaTail, anteriormente aislados, ahora están integrados en una interfaz fluida que guía a los usuarios a través del proceso de configuración y evaluación. La interfaz no solo ilustra la implementación, sino que también proporciona una herramienta interactiva para explorar las implicancias de distintas decisiones de configuración, demostrado en las imágenes ejemplo utilizadas en la sección 4.

5.2.2. Visualización y Comprensión

Al integrar visualizaciones directas de las configuraciones, la interfaz gráfica sirve como un puente entre la teoría y la práctica, permitiendo a los usuarios no solo leer sobre las configuraciones posibles, sino también interactuar con ellas. Esto enriquece la comprensión de los usuarios sobre cómo las diversas configuraciones afectan el proceso general, alineando la teoría presentada en la propuesta de DynaTail [39] con la aplicación práctica en la herramienta.

Así, la interfaz gráfica no solo sirve como un medio para la interacción del usuario sino también como un conector importante entre los distintos módulos de DynaTail. Esto ya que la interfaz y la herramienta presentan secciones dónde es posible integrar estos módulos a medida que se van desarrollando realizando cambios mínimos a la lógica de la aplicación. Con la implementación del modelador de contextos, y la integración del inyector de procesos, además de la visualización del grafo de influencias, ya hay tres componentes esenciales para la ejecución del método DynaTail de acuerdo a su propuesta [35]. Además, al quedar implementadas funciones con lógica simulada para la evaluación y optimización del proceso, se deja una base sólida sobre la cuál se pueden integrar los componentes faltantes del método, cómo las reglas del tailoring y la evaluación del proceso modificado utilizando el grafo de influencias. Esto presenta un avance considerable en la realización de DynaTail cómo un método aplicable en la industria del desarrollo de software, una vez que este método fue propuesto recientemente y no tenía un desarrollo que permitiera realizar su flujo y coordinar sus distintas partes, ya que aunque tenga partes con lógica simulada, la coordinación entre todas las partes desarrolladas de forma independiente es clave para el avance del método.

6. Conclusiones

En el presente trabajo de título, se ha desarrollado e implementado un modelador de contexto y una interfaz gráfica que sirva de base para la evaluación de procesos híbridos de desarrollo de software basándose en el método DynaTail, sirviendo cómo un primer paso para su realización.

Así, como los objetivos consistían en ofrecer una interfaz gráfica intuitiva y un módulo de transformación de modelos de contexto para implementar DynaTail, y se demostró que el modelador de contexto es capaz de generar modelos de contexto de proyectos a partir de los atributos definidos en un modelo de contexto organizacional, mientras que la interfaz gráfica proporciona una manera intuitiva y eficiente de interactuar con el método DynaTail, articulando sus distintos módulos, se puede concluir que se lograron cumplir en su totalidad los objetivos definidos para el trabajo.

Aún así, el cumplimiento de los objetivos tuvo complejidades y desafíos inherentes a la construcción de una aplicación desde cero. Si bien se presentaron dificultades, como la implementación de la visualización de procesos en formato BPMN, estas presentaron valiosas oportunidades para explorar soluciones de diseño y programación más profundas y creativas. Estos retos, aunque requirieron tiempo y esfuerzo adicionales, especialmente en la compatibilidad entre librerías, TypeScript y SvelteKit, han contribuido significativamente al enriquecimiento de la aplicación. El resultado es una herramienta funcional y eficiente, con un diseño limpio, que refleja la dedicación y el esfuerzo realizado en su desarrollo.

Con respecto a las tecnologías utilizadas, si bien la integración de SvelteKit, un framework no tan popular, con TypeScript y distintas librerías externas tuvo complicaciones que eventualmente lograron ser superadas, sus beneficios superaron los eventuales problemas que surgieron, ya que su capacidad de ejecutar código en el servidor desechó la necesidad de un proyecto aparte para la implementación de un 'backend'.

Además, el desarrollo realizado abre la puerta para el desarrollo de aún más funcionalidades, cómo por ejemplo el poder guardar y cargar los proyectos a medida que estos se fueran realizando.

De un punto de vista más externo, este trabajo contribuye significativamente al campo de la ingeniería de software, proporcionando una herramienta que facilita la adaptación y evaluación de procesos de desarrollo en contextos específicos, lo cual es especialmente relevante en la industria del software actual, donde los requerimientos y condiciones de proyectos varían ampliamente.

En conclusión, este trabajo de título representa un avance significativo en la implementación del método DynaTail, y por ende, en la gestión de procesos de desarrollo de software, sirviendo cómo una base sólida para coordinar los distintos componentes de este método a medida que estos sean desarrollados de forma independiente.

6.1. Trabajos Futuros

Como trabajos futuros para el desarrollo de esta aplicación, se considera lo siguiente:

1. Integrar los módulos faltantes del método DynaTail a la aplicación, como la evaluación del proceso, que actualmente no se encuentra completamente automatizada.
2. Mejorar la retroalimentación de la herramienta al realizar acciones, mostrándole al usuario directamente en la interfaz que sus acciones se ejecutaron de forma exitosa.
3. Implementar el guardado y carga de proyectos, permitiéndole a los usuarios retomar proyectos que ya hayan comenzado a adaptar en algún momento.
4. Permitirle al usuario descargar el proceso final adaptado una vez el atributo evaluado tenga un valor sastisfactorio, esto es, integrar el extractor desarrollado por Andres [**?**] al flujo de la aplicación.

7. Bibliografía

[1] Gustav Aagesen and John Krogstie. Bpmn 2.0 for modeling business processes. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, pages 219–250, 04 2015.

[2] Scott W. Ambler and Mark Lines. The disciplined agile process decision framework. In *International Conference on Software Quality. Process Automation in Software Development*, 2016.

[3] Pablo Berganza. Felte: A form library for svelte, solid and react. `https://github.com/pablo-abc/felte`. Visto por última vez el 14 de diciembre del 2023.

[4] Jean-Louis Boulanger. 5 - modeling. In Jean-Louis Boulanger, editor, *Certifiable Software Applications 3*. Elsevier, 2018.

[5] Camunda. A bpmn 2.0 rendering toolkit and web modeler. `https://github.com/bpmn-io/bpmn-js`. Visto por última vez el 14 de diciembre del 2023.

[6] Paul Clarke and Rory V O'Connor. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54(5):433–447, 2012.

[7] Philipp Diebold and Thomas Zeher. The Right Degree of Agility in Rich Processes. In *Managing Software Process Evolution*, pages 15–37. Springer, 2016.

[8] Philipp Diebold and Thomas Zehler. The agile practices impact model: idea, concept, and application scenario. In *Proceedings of the 2015 International Conference on Software and System Process, ICSSP'2015,*, pages 92–96. ACM, 2015.

[9] Philipp Diebold and Thomas Zehler. The agile practices impact model: Idea, concept, and application scenario. In *Proceedings of the 2015 International Conference on Software and System Process*, ICSSP 2015, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Peter Feiler and Watts Humphrey. Software process development and enactment: concepts and definitions. *Technical Report cmu/sei-92-tr-004, Software Engineering Institute*, 1992.

[11] Vicente García Díaz, Edward Núñez Valdez, Jordán Espada, B. Pelayo García-Bustelo, Juan Cueva Lovelle, and Carlos Marín. A brief introduction to model-driven engineering. 18:127–142, 04 2014.

[12] Asif Qumer Gill, Brian Henderson-Sellers, and Mahmood Niazi. Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. *Information Systems Frontiers*, 20:315–341, 2018.

[13] Ayşe Günsel, Atif Açikgšz, Ayça Tükel, and Emine Öğüt. The role of flexibility on software development performance: An empirical study on software development teams.

Procedia - Social and Behavioral Sciences, 58:853–860, 10 2012.

[14] Watts S Humphrey. The software engineering process: definition and scope. In *Proceedings of the 4th international software process workshop on Representing and enacting the software process*, pages 82–83, 1988.

[15] Andrew Hunt and David Thomas. *The Pragmatic programmer : from journeyman to master*. Addison-Wesley, Boston [etc.], 2000.

[16] Julio Hurtado, M. Bastarrica, Alcides Quispe, and Sergio Ochoa. Mde-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26, 04 2014.

[17] Julio Hurtado, Sergio Ochoa, Alcides Quispe, and María Bastarrica. A context modeling language to support tailoring of software processes. *Departamento de Ciencias de la Computación, Universidad de Chile, Chile*, Diciembre 2011.

[18] Jil Klünder, Regina Hebig, Paolo Tell, Marco Kuhrmann, Joyce Nakatumba-Nabende, Rogardt Heldal, Stephan Krusche, Masud Fazal-Baqaie, Michael Felderer, Marcela Fabiana Genero Bocco, et al. Catching up with method and process practice: An industry-informed baseline for researchers. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 255–264. IEEE, IEEE / ACM, 2019.

[19] Jil Klünder, Dzejlana Karajic, Paolo Tell, Oliver Karras, Christian Münkel, Jürgen Münch, Stephen MacDonell, Regina Hebig, and Marco Kuhrmann. *Determining context factors for hybrid development methods with trained models*. 01 2021.

[20] Marek Kubica. node-xml2js. `https://github.com/Leonidas-from-XIV/node-xml2js`. Visto por última vez el 14 de diciembre del 2023.

[21] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trektere, Fergal McCaffery, Oliver Linssen, Eckhart Hanser, and Christian R. Prause. Hybrid software and system development in practice: Waterfall, scrum, and beyond. In *Proceedings of the 2017 International Conference on Software and System Process*, ICSSP 2017, page 30–39, New York, NY, USA, 2017. Association for Computing Machinery.

[22] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Kitija Trektere, Fergal Mccaffery, Vahid Garousi, Michael Felderer, Oliver Linssen, Eckhart Hanser, and Christian Prause. Hybrid software development approaches in practice: A european perspective. *IEEE Software*, PP, 01 2018.

[23] Marco Kuhrmann, Paolo Tell, Jil Klünder, Regina Hebig, Sherlock Licorish, and Stephen MacDonell. Complementing materials for the helena study (stage 2). *DOI: 10.13140/RG.2.2.11032.65288*, Noviembre 2018.

[24] Steffen Küpper, Dietmar Pfahl, Kristjan Jürisoo, Philipp Diebold, Jürgen Münch, and Marco Kuhrmann. How has SPI changed in times of agile development? results from a multi-method study. *Journal of Software Evolution and Process*, 31(11), 2019.

[25] UUID JavaScript Module. Uuid. `https://github.com/uuidjs/uuid`. Visto por última vez el 14 de diciembre del 2023.

[26] Jürgen Münch, Ove Armbrust, Martin Kowalcyzk, and Martín Soto. *Software Process Definition and Management*. Springer-Verlag, Germany, 2012.

[27] Daniel Antonio Ortega Norambuena. Diseño e implementación de herramienta para la configuración de contextos en proyectos de software. *Departamento de Ciencias de la Computación, Universidad de Chile, Chile*, Agosto 2012.

[28] Observable. D3: Data-driven documents. `https://github.com/d3/d3`. Visto por última vez el 14 de diciembre del 2023.

[29] Oscar Pedreira, Mario Piattini, Miguel Luaces, and Nieves Brisaboa. A systematic review of software process tailoring. *ACM SIGSOFT Software Engineering Notes*, 32:1–6, 05 2007.

[30] Nils Prenner, Carolin Unger-Windeler, and Kurt Schneider. Goals and challenges in hybrid software development approaches. *Journal of Software: Evolution and Process*, 33(11):e2382, 2021.

[31] Teguh Raharjo and Betty Purwandari. Agile Project Management Challenges and Mapping Solutions: A Systematic Literature Review. In *Proceedings of the 3rd International Conference on Software Engineering and Information Management, ICSIM '2020*, page 123–129, New York, NY, USA, 2020. ACM.

[32] Md Shamsur Rahim, AZM Chowdhury, Dip Nandi, Mashiour Rahman, and Shahadatul Hakim. Scrumfall: A hybrid software process model. *International Journal of Information Technology and Computer Science*, 10:41–48, 12 2018.

[33] Open Source. Lodash. `https://github.com/lodash/lodash`. Visto por última vez el 14 de diciembre del 2023.

[34] Open Source. Prettier. `https://github.com/prettier/prettier`. Visto por última vez el 14 de diciembre del 2023.

[35] Jacqueline Sánchez, M. Bastarrica, Julio Hurtado, and Luis Silvestre. Dynatail: A method for hybrid software process tailoring. *Departamento de Ciencias de la Computación, Universidad de Chile - Chile, IDIS Research Group, Universidad del Cauca - Colombia, Departamento de Ciencias de la Computación, Universidad de Talca - Chile*.

[36] Paolo Tell, Jil Klünder, Steffen Küpper, David Raffo, Stephen G. MacDonell, Jürgen Münch, Dietmar Pfahl, Oliver Linssen, and Marco Kuhrmann. What are hybrid development methods made of?: an evidence-based characterization. In Stanley M. Sutton Jr., Ove Armbrust, and Regina Hebig, editors, *Proceedings of the International Conference on Software and System Processes, ICSSP 2019, Montreal, QC, Canada, May 25-26, 2019*, pages 105–114. IEEE / ACM, 2019.

[37] Michael Unterkalmsteiner, Tony Gorschek, A.K.M. Moinul Islam, Chow Kian Cheng,

Rahadian Bayu Permadi, and Robert Feldt. Evaluation and measurement of software process improvement—a systematic literature review. *IEEE Transactions on Software Engineering*, 38(2):398–424, 2012.

[38] Leo Vijayasarathy and Charles Butler. Choice of software development methodologies - do project, team and organizational characteristics matter? *IEEE Software*, 1:1–1, 01 2015.

[39] Andrés Wallberg, Daniel González, Luis Silvestre, and María Cecilia Bastarrica. A tool for modeling and tailoring hybrid software processes. In *12th International Conference on Model-Based Software and Systems Engineering (submitted)*, 2024.

[40] Andrés Wallberg and Luis Silvestre. Injector and extractor for transformations between business processes and business process models. In *2023 42nd IEEE International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–8, 2023.

[41] Michael Weiss. *XML Metadata Interchange*, pages 3597–3597. Springer US, Boston, MA, 2009.

[42] Włodzimierz Wysocki. A hybrid software processes management support model. *Procedia Computer Science*, 176:2312–2321, 01 2020.

Anexos

A. Anexo A: Archivo XMI. Resultado de la generación por parte de la aplicación de un modelo de contexto de un proyecto

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<spcm:Context xmlns:spcm="http://contextmetamodel/1.0" xmlns:xmi="http://www.omg.org/XMI" description="Test" name="Test" xmi:id="ee9be64c-a527-
    <myDimensions description="" name="Project" xmi:id="4267c21b-42bc-4500-a6a6-52d9d1f9103a">
        <myContextAttributes description="" name="Project type" xmi:id="3e726f83-ccee-435a-868e-7cfc1cd1496a">
            <posibleValues description="New" name="New" value="New" xmi:id="b55fdb22-5023-4418-9bd4-d945b62604f5"/>
            <posibleValues description="Old" name="Old" value="Old" xmi:id="5262f0b1-78c3-4c65-a089-5e08a4b02ecb"/>
        </myContextAttributes>
    </myDimensions>
    <myDimensions description="" name="Team" xmi:id="64191ddb-2c6c-48e3-b364-797dba2674dd">
        <myContextAttributes description="" name="Technology knowledge" xmi:id="345d2538-6668-4110-a41d-5d7cd108c2bf">
            <posibleValues description="High" name="High" value="High" xmi:id="6e23f8fa-649a-42fa-b1c3-5f69dd4a5208"/>
            <posibleValues description="Average" name="Average" value="Average" xmi:id="89e28124-4842-4ae9-b5b5-0ddb9e188f3e"/>
            <posibleValues description="Low" name="Low" value="Low" xmi:id="f7ff0087-1b2f-473a-9d56-d53512eaa187"/>
        </myContextAttributes>
        <myContextAttributes description="" name="Domain knowledge" xmi:id="8bef5ef8-1d34-40b1-8f17-84b259612304">
            <posibleValues description="High" name="High" value="High" xmi:id="f8ade533-e572-414b-8b19-4a415c1293e3"/>
            <posibleValues description="Average" name="Average" value="Average" xmi:id="6986bc56-076d-4527-861c-ae632b945c16"/>
            <posibleValues description="Low" name="Low" value="Low" xmi:id="97f2b523-56ad-4655-8a29-fb79156f7a8b"/>
        </myContextAttributes>
        <myContextAttributes description="" name="Team size" xmi:id="653a8ed9-d7d8-4b13-8721-c1f8887d32c2">
            <posibleValues description="Large" name="Large" value="Large" xmi:id="a3f5ea66-7317-4a99-99f0-4116ba0d2b7a"/>
            <posibleValues description="Average" name="Average" value="Average" xmi:id="487c2094-10a0-46fa-ab1a-73ca8ce12610"/>
            <posibleValues description="Small" name="Small" value="Small" xmi:id="5bb6f685-aefc-43fc-aa98-ee23afcdfaaf"/>
        </myContextAttributes>
    </myDimensions>
    <myDimensions description="" name="Product" xmi:id="6597905a-84d3-4047-bb84-53cf4afa874a">
        <myContextAttributes description="" name="Usability" xmi:id="b431dfae-1c51-436a-bdaa-4070aa784aaa">
            <posibleValues description="High" name="High" value="High" xmi:id="bb7f3a43-a29e-4efb-8d91-5e79fbbd5722"/>
            <posibleValues description="Low" name="Low" value="Low" xmi:id="ae6b426c-45d9-44be-b739-79e833a0b221"/>
        </myContextAttributes>
    </myDimensions>
    <myContextConfigurations name="Context model" xmi:id="78065f2e-247a-4628-a129-7b01a293dcc9">
        <contextAttributeConfiguration description="" myContextAttributeValue="5262f0b1-78c3-4c65-a089-5e08a4b02ecb" myContextElement="3e726f8
        <contextAttributeConfiguration description="" myContextAttributeValue="89e28124-4842-4ae9-b5b5-0ddb9e188f3e" myContextElement="345d253
        <contextAttributeConfiguration description="" myContextAttributeValue="6986bc56-076d-4527-861c-ae632b945c16" myContextElement="8bef5ef
        <contextAttributeConfiguration description="" myContextAttributeValue="5bb6f685-aefc-43fc-aa98-ee23afcdfaaf" myContextElement="653a8ed
        <contextAttributeConfiguration description="" myContextAttributeValue="ae6b426c-45d9-44be-b739-79e833a0b221" myContextElement="b431dfa
    </myContextConfigurations>
</spcm:Context>
```


Figura 21: Modelo de contexto de un proyecto

B. Anexo B: Interfaces desarrolladas para la aplicación

```typescript
export interface Node {
  type: string;
  name: string;
}

export interface Influence {
  source: number;
  target: number;
  value: number;
}

export interface InfluenceGraph {
  nodes: Node[];
  influences: Influence[];
}

export interface Context {
  dimensions: Dimension[];
}

export interface Dimension {
  name: string;
  attributes: Attribute[];
}

export interface Attribute {
  name: string;
  options: string[];
}
```

Figura 22: Interfaces implementadas en la aplicación

C. Anexo C: Archivo BPMN. Proceso BPMN utilizado como input al inyector

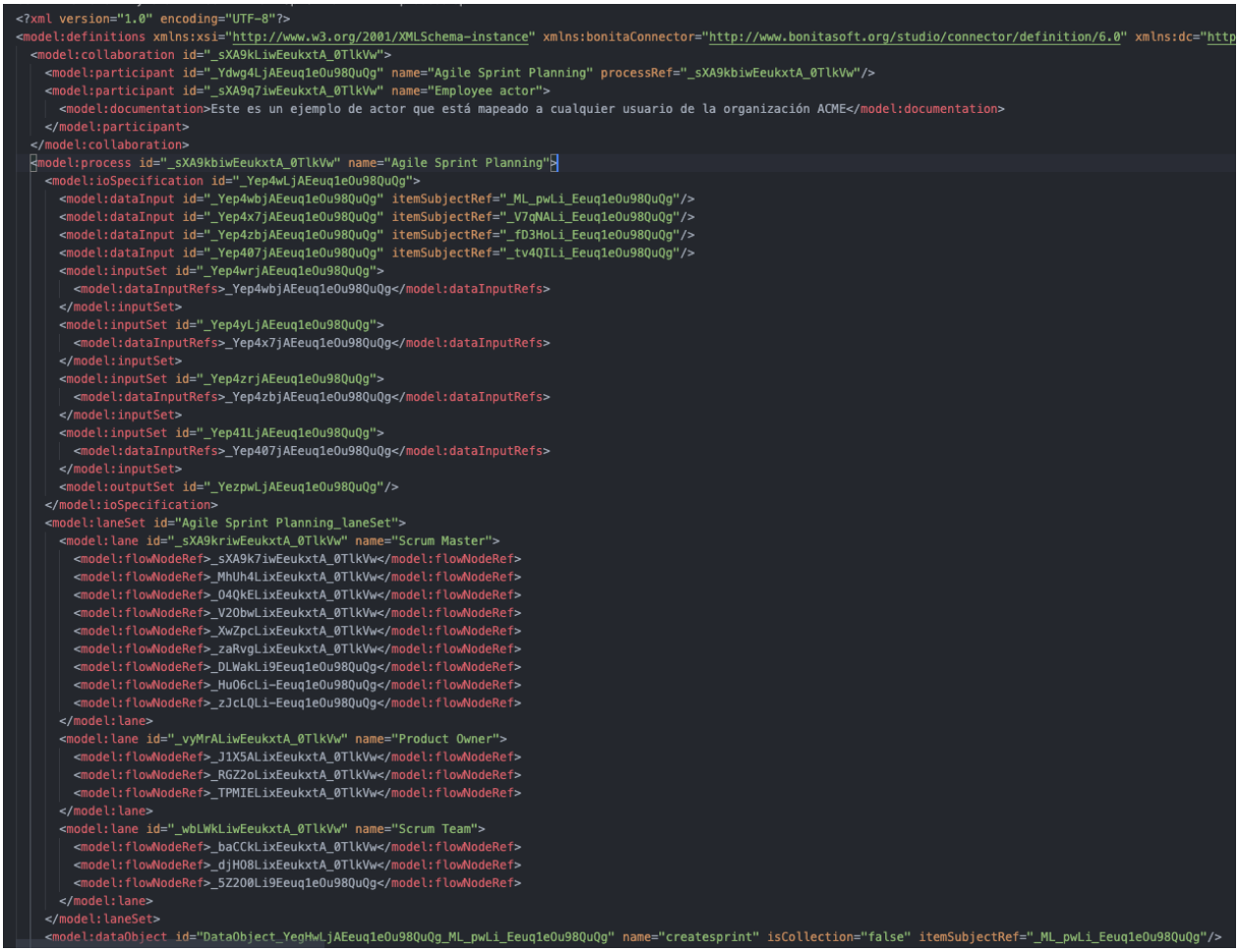


Figura 23: Proceso BPMN utilizado de input para la aplicación

D. Anexo D: Archivo XMI. Modelo de proceso BPMN resultado de aplicar el inyector al proceso de la figura 23

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:Definitions xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XML
    <rootElements xsi:type="bpmn2:Process" name="Process">
        <documentation text=""/>
        <flowElements xsi:type="bpmn2:DataObject" name="createsprint isCollection=">
        </flowElements>
        <flowElements xsi:type="bpmn2:DataObject" name="newstories isCollection=">
        </flowElements>
        <flowElements xsi:type="bpmn2:DataObject" name="morestories isCollection=">
        </flowElements>
        <flowElements xsi:type="bpmn2:DataObject" name="consensus isCollection=">
        </flowElements>
        <flowElements xsi:type="bpmn2:StartEvent">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Create Sprint">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Ask for new stories">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Select story">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Estimate sprint story">
        </flowElements>
        <flowElements xsi:type="bpmn2:EndEvent">
        </flowElements>
        <flowElements xsi:type="bpmn2:ExclusiveGateway" name="_DLWakLi9Eeuq1eOu98QuQg">
        </flowElements>
        <flowElements xsi:type="bpmn2:ExclusiveGateway" name="_HuO6cLi-Eeuq1eOu98QuQg">
        </flowElements>
        <flowElements xsi:type="bpmn2:ExclusiveGateway" name="_zJcLQLi-Eeuq1eOu98QuQg">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Select Stories">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Provide stories">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Prioritize stories">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Estimate tasks">
        </flowElements>
        <flowElements xsi:type="bpmn2:Task" name="Converge to consensus">
        </flowElements>
        <flowElements xsi:type="bpmn2:ExclusiveGateway" name="_5Z2O0Li9Eeuq1eOu98QuQg">
        </flowElements>
    </rootElements>
</bpmn2:Definitions>
```

Figura 24: Modelo de proceso BPMN producto del inyector

E. Anexo E: Archivo XMI. Grafo de influencias utilizado como ejemplo

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dynatail:CGModel xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http:
  <cgnode xsi:type="dynatail:CGCharacteristic" name="Customer Value"/>
  <cgnode xsi:type="dynatail:CGAttribute" name="BacklogQuality"/>
  <cgnode xsi:type="dynatail:CGActivity" name="EstimateTask"/>
  <cgnode xsi:type="dynatail:CGPractice" name="Team-based Estimation" description=""/>
  <cginfluence source="//@cgnode.3" target="//@cgnode.2" value="2.0"/>
  <cginfluence source="//@cgnode.2" target="//@cgnode.1" value="0.6"/>
  <cginfluence source="//@cgnode.1" target="//@cgnode.0" value="0.7"/>
</dynatail:CGModel>
```

Figura 25: Grafo de influencias en formato XMI

F. Anexo F: Archivo XMI. Modelo de contexto de un proyecto externo a la herramienta

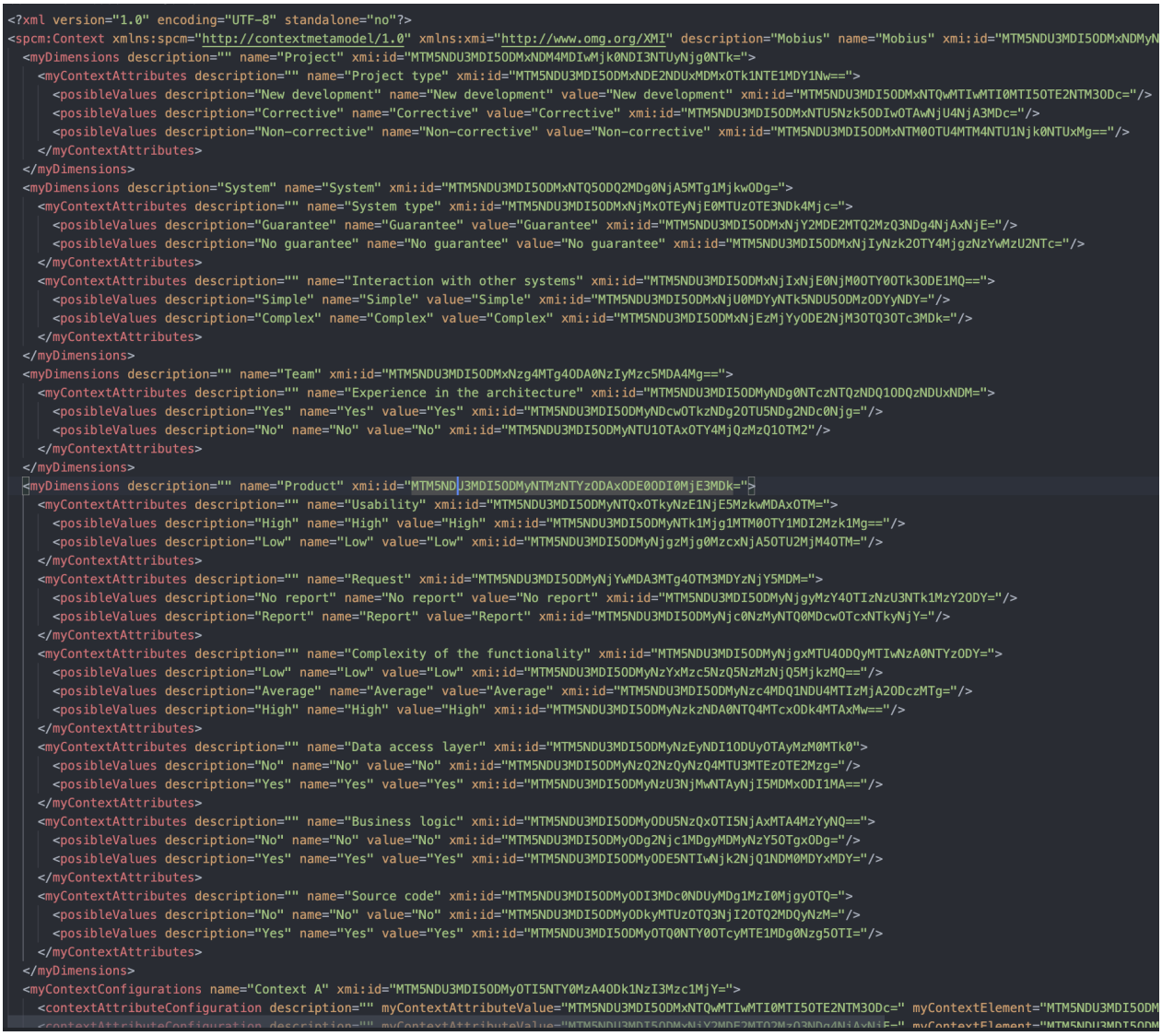


Figura 26: Modelo de contexto de un proyecto generado y validado de forma externa a la herramienta

G. Anexo G: Archivo XMI. Modelo de contexto de un proyecto generado por la herramienta en base a la Figura 26

Figura 27: Modelo de contexto de un proyecto generado por la herramienta, en base al modelo del anexo F