Desarrollo de una línea de productos de software para aplicaciones Feed Mashup

Héctor Reinaga[1], Juan Enriquez[1], Sandra Casas[1]
[1] Instituto de Tecnología Aplicada, Universidad Nacional de la Patagonia Austral (UARG) - Av. Gregores y Piloto Lero Rivera - Río Gallegos, Santa Cruz - Argentina

{hreinaga, jenriquez, scasas}@uarg.unpa.edu.ar

Resumen. Las aplicaciones Mashup pueden ser consideradas como una tendencia en el desarrollo de software durante los últimos años, porque integra dos o más tipos de componentes disponibles en la Web. Las fuentes o feeds RSS se consideran también como un componente Mashup entre otras tecnologías. Una Líneas de Productos de Software (LPS) es un enfoque de desarrollo de software cuyo principal objetivo es la reusabilidad, permitiendo crear una familia de productos donde cada producto posee características comunes, y difiere de otro en un conjunto de funcionalidades. Las actuales herramientas y enfoques de desarrollo de las aplicaciones mashup carecen de técnicas, métodos, y enfoques para su tratamiento. Precisamente, en búsqueda de métodos y/o herramientas que permitan construir aplicaciones, que no requiera de conocimiento en lenguaje específico por parte del usuario, que admitan reusabilidad y flexibilidad, como así genere código automático; este trabajo propuso un enfoque para modelar, diseñar e implementar una aplicación Mashup desde una perspectiva de variabilidad, lo cual permitió crear una LPS para este dominio. Como resultado se presentó un modelo de características abstracto para la generación automática de aplicaciones Feed Mashup, y una herramienta que da soporte al modelo.

Palabras claves: Mashup, Línea de Producto de Software, Modelo de Características, Feature, Feeds.

1 Introducción

Las aplicaciones Mashup se consideran como una tendencia en el desarrollo de software durante los últimos años, porque integra al menos dos componentes disponibles en la Web, creando un nuevo valor, así permite el reuso y proporciona una funcionalidad que no existía antes [1]. Un componente es cualquier segmento de datos, lógica de aplicación y/o interfaz de usuario que puede ser reutilizada y que es accesible ya sea local o remotamente [2]; y se basan en lenguajes estándar, tecnologías o protocolos de comunicación.

El crecimiento de la información digital requiere de métodos eficientes para administrar y filtrar datos no deseados a través de la Web. Una herramienta utilizada por la industria y los usuarios para recopilar información valiosa de Internet son los deno-

minados feed o fuentes RSS [3]. Básicamente, los feeds RSS recopilan contenido nuevo de los sitios web visitados con más frecuencia, y lo distribuyen a un programa o agregador de noticias, navegador o cuenta de correo electrónico que posea el usuario [4]. RSS es un formato basado en XML para distribuir y agregar contenido web. RSS es una de las innovaciones más populares en Internet que resuelve el problema de la recopilación y distribución de la información disponible en la Web, logrando organizar la información generada por millones de sitios web que publican contenido a diario. Las fuentes RSS se consideran un componente Mashup entre otras tecnologías [5], y en consecuencia, no escapa a las dificultades que se encuentran en el proceso de desarrollo de los mismos, debido al volumen y heterogeneidad de componentes mashup disponibles en la Web; la ausencia de algún modelo para integrar componentes similares; la selección de los modelos adecuados para la integración.

Una Línea de Productos de Software es un enfoque de desarrollo de software cuyo principal objetivo es la reusabilidad, permitiendo crear una familia de productos, que comparten un conjunto común de características, que satisface las necesidades específicas de un dominio o segmento de mercado en particular [6], y difiere de otro en un conjunto de funcionalidades opcionales (variables). Esta diferencia funcional entre productos de una LPS se conoce como variabilidad [7]. Para expresar características comunes se crean Modelos de Características (MC) y para manejar la variabilidad entre los productos de una LPS, Modelos de Variabilidad [8] [9].

Por lo tanto, y en búsqueda de métodos y/o herramientas que permitan construir aplicaciones, que no requieran de conocimiento en lenguajes específico por parte del usuario, que admitan reusabilidad y flexibilidad, como así generen código automáticamente; este trabajo formula un modelo conceptual en base a un enfoque de variabilidad (características) y LPS. Así como resultado, se realizó su implementación a través de una herramienta para la generación automática de aplicaciones Feed Mashup.

El resto de este documento se organiza de la siguiente manera. La Sección 2 analiza los trabajos relacionados. La Sección 3 describe los enfoques y estrategias utilizados. La Sección 4 presenta un MC para aplicaciones Feed Mashup. La Sección 5 analiza la herramienta LPS-FeedMashup que implementa el modelo. La Sección 6 presenta las evaluaciones. La Sección 7 presenta las conclusiones y los trabajos futuros.

2 Trabajos relacionados

Se propusieron diversos enfoques para el desarrollo de Mashup; seguidamente, se describen algunos de ellos:

En [10] se presentó un estudio a través de la selección de API abiertas híbridas para desarrollar Mashups con el fin que los desarrolladores encuentren de manera eficiente y precisa, las API abiertas que necesitan, mediante un enfoque de descubrimiento y recomendación. En [11] se describió un enfoque de desarrollo de mashup para la recomendación del paquete de servicios utilizando minería de descripción de texto. Evaluaron la propuesta con datos del mundo real, los resultados demostraron que el estudio propuesto es capaz de lograr un mayor nivel de precisión, y también tiene mejores resultados en comparación con otros enfoques. En [12] se presentó una

revisión sistemática de herramientas, lenguajes y metodologías utilizadas para desarrollar mashups. El propósito es ayudar a las empresas a elegir las mejores herramientas y métodos, como también a los investigadores, para comprender el estado actual de las herramientas de desarrollo de mashup. En [13] se presentó un enfoque en el campo del desarrollo de mashups para la recomendación de API, y se basa en un modelo probabilístico que recomienda una lista de APIs que se pueden usar para componer cualquier mashup, proporcionando descripciones y detalles del mismo. En [14] se propuso un método innovador llamado "mash light" para la creación de mashup utilizando widgets Web 2.0; Mash light es liviano y puede ejecutarse en el navegador sin ningún requisito de servidor, así también, se enfatizó la fácil composición de los servicios web sin experiencia técnica. En [15] también se presentó un enfoque de mashup basado en la Web para una mejor usabilidad y satisfacción del usuario, integrando múltiples fuentes web. En el estudio, el modelo de componentes separa dinámicamente el servicio comercial y la interfaz del usuario, y también proporciona un modelo de prototipo basado en navegador. En [16] se describió un estudio sobre un sistema que ahorra tiempo y esfuerzo, aumentando la facilidad de uso al comprender la situación y el contexto del usuario, y luego crear widgets para mashup sin ninguna configuración manual. El sistema proporcionó dinámicamente widgets de mashup para crear mashup en tiempo de ejecución al estar atento a las necesidades del usuario. En [17] se propuso un método de recomendación de API basado en la técnica de filtrado colaborativo de gráficos neuronales, que aprovecha las interacciones históricas entre el usuario y las API, y así obtener un sistema de recomendación de API de alta calidad.

3 Enfoques y estrategias

3.1 Línea de productos de software y modelo de características

Una LPS es un paradigma de desarrollo de software que busca crear familias de productos en lugar de crear productos individuales. Estas familias agrupan un conjunto de productos que comparten características comunes a la vez que tienen algunas variaciones entre sí. La LPS posee beneficios tales como la reutilización, disminución de los tiempos y costos de desarrollo [18].

Una LPS se representa por medio de modelos. Una de las técnicas que existen para expresar estos modelos se denomina modelos de características [19]. Una "característica" (feature) es un rasgo o elemento distintivo que representa aspectos relevantes de un dominio de aplicación según el punto de vista del usuario o desarrollador. Las características se usan en una LPS para especificar y comunicar aspectos comunes y variables de la LPS [20]. Las características se relacionan entre sí con dependencias, entonces un modelo jerárquico puede ser creado, clasificando y estructurando las características, utilizando distintos tipos de relaciones. El método original [21] clasifica las características en obligatorias, opcionales o alternativas. El modelo se completa, con la especificación de las restricciones, que se conforma como un conjunto de reglas (expresiones lógicas formadas por características, conectivos lógicos y cuantificadores) [22]. Varias herramientas [23] apoyan el desarrollo de software orientado a

características. Entre estos, se utiliza FeatureIDE [24] como marco para el presente estudio. La Fig. 1 presenta un MC representado con la herramienta FeatureIDE. Los nodos MC representan entidades (características) y las líneas indican las relaciones entre ellas. El nodo raíz A, representa el concepto de dominio que se está modelando.

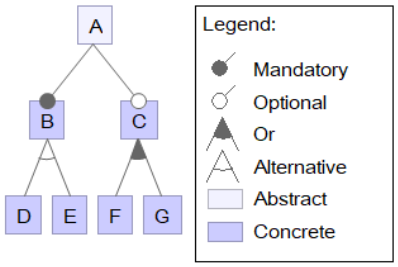

Fig. 1. Representación de modelo de características.

Los MC se han aplicado en un amplio rango de dominios, y en particular el desarrollo de LPS en distintos dominios, como telefonía móvil [25], robótica, geográficos [26], TV Digital [27], dashboards o paneles de información en contextos educativos [28], entre otros; no encontrándose así para el dominio de aplicaciones Mashup.

3.2 Mashup y Feed RSS

Las aplicaciones mashup pueden incluir diversos componentes disponibles en la Web, entre ellos las fuentes o feed RSS. Se han propuesto una variedad de herramientas para el desarrollo de aplicaciones Feed Mashup. A continuación, se describen algunas de ellas:

Yahoo! Pipes fue una herramienta para la manipulación de los feeds, creada en el año 2007 por la empresa Yahoo!, y se convirtió en una de las más utilizadas y populares en su momento, hasta el año 2015 que fue finalizado el proyecto. RSS Mix (http://www.rssmix.com/) es una herramienta para mezclar o combinar hasta 100 feeds RSS en un único feed, puede generar feeds en formato JSON para usarlo en otras aplicaciones. Feed Informer (http://feed.informer.com/) permite mezclar y convertir feeds de manera rápida, requiere el registro mediante una dirección de correo electrónico, y se puede elegir distintas opciones de salida (HTML a PDF), personalizar la plantilla de feed, agregar y publicar un resumen del mismo. RSS Mixer (https://rssmixer.com/) es una forma sencilla de combinar varios canales RSS en un solo canal, la versión gratis es limitada, ofrece a los usuarios una solución rápida y simple para mezclar sus feeds de manera inmediata. Finalmente, Pipes.Digital (https://www.pipes.digital), es el nombre de un servicio que permite combinar varias fuentes de información utilizando los feeds RSS o ATOM, y aplicar filtros para que el resultado sea otro feed con los criterios que se definan; funciona de manera similar a Yahoo! Pipes.

El enfoque metodológico aplicado al estudio, corresponde al DSR (Design Science Research) [29] [30], y se dirige a la producción de artefactos. Precisamente, en búsqueda de métodos y/o herramientas que permitan construir aplicaciones que admitan

reusabilidad y flexibilidad, y habiendo analizado algunas herramientas de desarrollo mashup descriptas anteriormente; el trabajo se enfocó en el entorno de desarrollo visual basado en Web para el usuario final, denominado Pipes Digital (PD), ya que posee características y funcionalidades que permite a los usuarios reunir datos y gestionar la información obtenida de diferentes fuentes, y procesarlos de manera intuitiva, como así también, funciona de manera similar a Yahoo! Pipes, uno de los servicios de gestión de contenidos feeds más utilizados en su momento. Por lo tanto, se procedió a la identificación y el análisis de las funcionalidades de dicho entorno de desarrollo para su posterior implementación en un modelo genérico abstracto.

PD consiste básicamente en un editor de programación visual para obtener datos de la Web a través de los feeds, y permite combinar bloques de datos (pipes) de manera intuitiva. En Tabla 1, se presentan las principales funcionalidades de PD.

Tabla 1. Funcionalidades principales de PD.

Nombre	Funciones	Objetivo
Input	Feed, Download, Webhook, Text input, y Pipe	Bloque principal para el ingreso de datos
Integrations	Dailymotion, Mixcloud, Periscope, Reddit, Speedrun, SVT Play, Tweets, Ustream, y Vimeo.	Permite integrar otras aplicaciones
Manipulate	Filter, Replace, Unique, Truncate, Sort, y Shorten	Realiza distintas operaciones, como filtrar, ordenar por criterio, y otras
Control	Combine, Duplicate, Merge Items, y Foreach	Permite combinar o duplicar varios feeds en uno solo, entre otras
Create	Extract, Images, Tables, Insert, y Build Feed	Extrae elementos de una página o feed, para crear un nuevo feed

4 Modelado de características para aplicaciones Feed Mashup

El MC consistió en analizar el dominio de la LPS, definir su alcance, identificar los elementos comunes y los componentes variables de la LPS, identificar las restricciones del modelo, y diseñar los aspectos reutilizables que soportan la LPS. El resultado se expresa en un diagrama de características, y un conjunto de reglas.

4.1 Modelo propuesto

En esta etapa se definió el alcance de la LPS, cada característica se representa como una funcionalidad de los bloques definidos por PD. Las operaciones relevantes se encuentran en los grupos de Input, Integration, Manipulate, Control, y Create.

Para obtener una primera representación del MC, se realizó un proceso iterativo de refinamiento. Así, inicialmente fue elaborado un MC para cada una de las características encontradas. Luego, cada uno de estos modelos fue examinado con el objetivo de identificar en ellos, características y dependencias comunes. Después se identificaron aquellas características que pudieran ser agrupadas en características más generales; y finalmente, se clasificaron en obligatorias (forman parte de todos los productos que contienen la LPS), opcionales (puede ser o no parte de un producto), alternativas

(sólo una característica puede ser seleccionada), y disyuntivas (más de una característica puede ser elegida).

A partir de este análisis se obtuvo un conjunto inicial de relaciones: 1) se definió la característica InputIntegration como característica obligatoria; 2) las características Manipulate, Control, y Create como características opcionales (abstractas).

Los resultados obtenidos permitieron diseñar el MC que se muestra en la Fig. 2. El diagrama indica las clasificaciones correspondientes a características obligatorias, opcionales y alternativas.

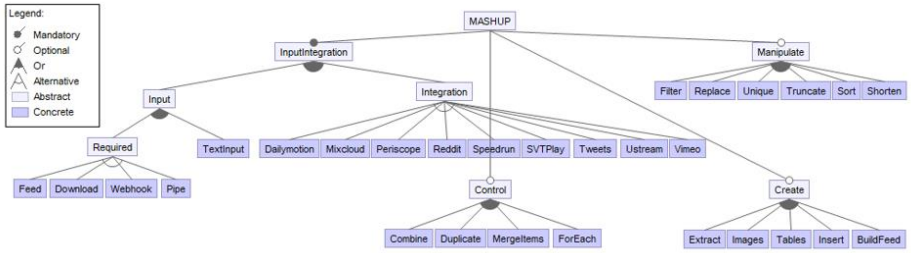

Fig. 2. Modelo de características de la LPS para aplicaciones Mashup.

4.2 Configuración de Mashup

Al modelo se agregó un conjunto de restricciones lógicas o fórmulas que deben cumplir las características. Además, estas reglas permiten depurar productos válidos y a su vez contribuyen a la trazabilidad del modelo creado.

La configuración permitió expresar en forma textual el MC. Estas reglas indican, que solo una de las características del grupo Integration debe ser seleccionada. Solo una de las características de InputIntegration puede ser seleccionada. Más de una de las características de Input pueden ser elegidas. Solo una de las características del grupo Required que depende de la característica Input, puede ser seleccionada. Más de una característica del grupo Manipulate, Control y Create pueden ser seleccionadas. Por último, la característica TextInput requiere al menos de una de características hijas que le dependen.

La instanciación de los productos mashup del MC deben cumplir con distintos tipos de limitaciones (restricciones del modelo), las que se enumeran a continuación:
R#1: Required implies not Integration
R#2: Pipe implies not TextInput
R#3: Extract and Images and Tables implies Feed or Download
R#4: Insert implies Feed and Extract
R#5: BuildFeed implies (Feed or Download) and Extract
R#6: MergeItems implies Feed and Extract
R#7: ForEach implies Feed or Download or Tweets

Seguidamente, se describen brevemente algunas de las reglas del MC: la regla #1 es una restricción que deshabilita el bloque Integration cuando se selecciona algunas de las características del grupo Required; la regla #2, deshabilita el bloque TextInput

cuando es elegido el bloque Pipe; y la regla #3, los bloques Extract, Images, y Tables solo pueden ser usados en bloques Feed o Download.

En resumen, el diagrama presentado en la Fig. 2, representa un total de 37 características, de las cuales 7 son abstractas y 30 concretas. En Tabla 2, se muestra el total de las relaciones obligatorias, opcionales, alternativas y disyuntivas, como así también, el total de características terminales o hijas.

Tabla 2. Funcionalidades principales del modelo.

Nombre	InputInte-gration	Manipulate	Control	Create	Feed-Mashup	Total
Obligatorias	1				1	2
Opcionales	2	1	1	1		5
Alternativas	13					13
Disyuntivas	2	6	4	5		17
Total	18	7	5	6	1	37
Terminales	14	6	4	5	1	30

5 Herramienta Feed Mashup: diseño e implementación

La herramienta LPS-FeedMashup implementó el modelo conceptual abstracto y genérico descripto anteriormente. Se realizó en lenguaje Java y FeatureIDE con el componente FeatureHouse [31].

El Diagrama de Clases de la Fig. 3 corresponde a las clases que componen a la herramienta. La misma está conformada por las clases Main (módulo principal), Feed, FeedMessage, RSSFeedOperation,RSSFeedWriter, y RSSFeedParser; por otro lado las clases Filter, Replace, Unique, Truncate, Sort, Shorten, Extract, Images, Tables, Insert, BuilFeed, Combine, Duplicate, MergeItems, ForEach, TextInput, Download, Webhook, Pipe, Tweets, Vimeo, SVTPlay, Mixcloud, Speedrun, Dailymotion, Periscope, Ustream, Reddit, Instruction, y Section, conforme las funcionalidades vistas. De acuerdo con este diagrama, a partir de la clase Main, se crean distintos objetos que permiten crear un producto mashup.

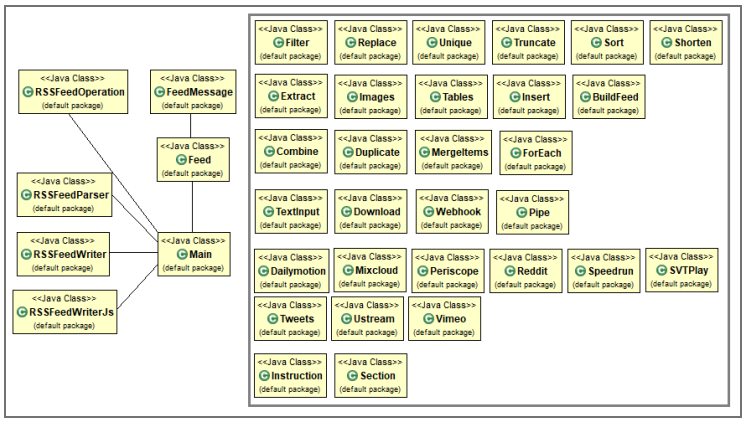

Fig. 3. Diagrama de Clases LPS-FeedMashup.

La herramienta propuesta genera un producto mashup en tres pasos:

1- Configuración de Mashup: El primer paso para la generación de un producto consiste en seleccionar las operaciones del mismo. Para ello, se deben especificar los elementos de interacción (nombre o URL feed, filtros, combinar, duplicar, etc.). La herramienta ofrece un menú con todas las funciones para realizar sobre un feed. Un producto de una LPS se especifica de forma declarativa seleccionando o anulando la selección de operaciones de acuerdo a la necesidad del usuario (Fig. 4). Es por esto que, las decisiones tomadas deben respetar las limitaciones del MC (parámetros iniciales). Como resultado se obtiene un archivo de configuración.

2- Generación de un Mashup: Después de seleccionada una determinada configuración, se genera automáticamente el código Javascript embebido dentro de un código HTML, que implementa las operaciones seleccionadas en el paso 1.

3- Generación de un Feed RSS: Asimismo, como complemento del paso anterior, la herramienta permite la creación de un archivo feed RSS, con las operaciones seleccionadas por el usuario.

La herramienta organiza la disposición de los archivos generados en diversas carpetas. Por cada producto genera una nueva carpeta con un nombre por defecto o especificado por el usuario.

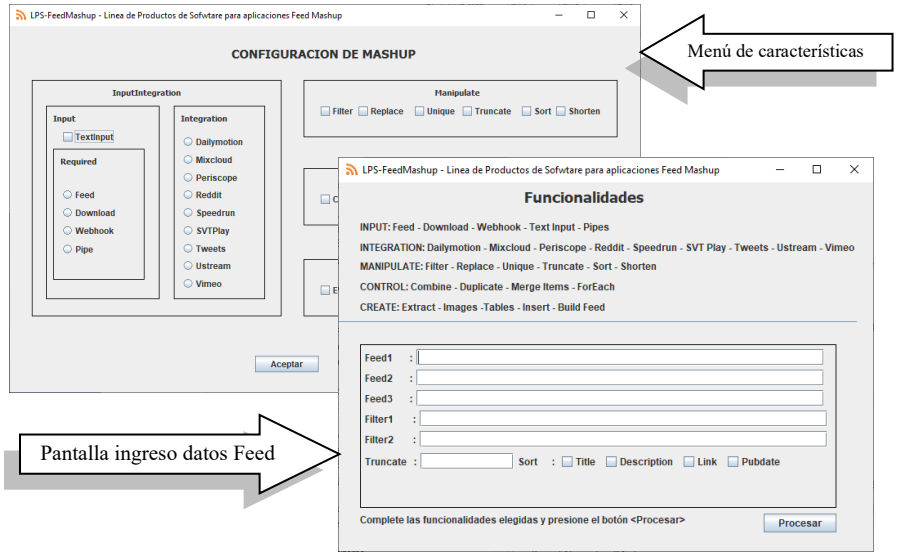

Fig. 4. Pantalla de selección e ingreso datos desde LPS-FeedMashup.

6 Evaluaciones

Se realizaron dos tipos de evaluaciones para esta propuesta: la experimentación con casos de estudio, y la evaluación de la calidad del MC.

6.1 Creación de prototipos, experimentación y caso de estudio

Esta parte del estudio consistió en utilizar la herramienta LPS-FeedMashup aplicada en dos casos de estudio. El primer caso, se refiere a una consulta a la revista IEEE Transactions on Software Engineering, por los términos Software Product Line, Patterns, y Quality. Y como segundo caso, se realizó una consulta sobre noticias de nuestro país, ordenado por fecha, en distintos sitios Web de relevancia, como France24 (Francia), The New York Times (EEUU), El País (España), e Infobae (Argentina).

La experimentación de estos casos tuvo varios objetivos: probar la correcta funcionalidad de la herramienta, verificar el diseño de diferentes configuraciones, comprobar que las restricciones sean correctas, comprobar que el código generado sea válido, y principalmente, demostrar que la herramienta se pueda utilizar para desarrollar productos de una manera automatizada y que genere código fuente sin requerir de conocimiento en lenguaje específico por parte del usuario.

En la Tabla 3 se especifica la información entrada para los casos de estudio.

Tabla 3. Correspondencia de entradas y características para el caso de estudio 1 y 2.

Caso de Estudio	Información de entrada	Grupo de características	Sub-características	Características
1	Ingreso feed con su URL	InputIntegration	Input-Required	Feed
	Divide feed para manipular	Control		Duplicate
	Filtra feed por palabra clave	Manipulate		Filter
	Combina los elementos de entrada en un solo feed	Control		Combine
2	Ingreso feed con su URL	InputIntegration	Input-Required	Feed
	Combina elementos de feed	Control		Combine
	Filtra feed por palabra clave	Manipulate		Filter
	Ordena feed por un elemento	Manipulate		Sort

La Fig. 5 muestra la interfaz de configuración de la herramienta para el caso 1. Las opciones elegidas se resaltan en la interfaz de la herramienta del siguiente modo: a los grupos de características con rectángulos, a las sub-características con rectángulos redondeados, y a las características con elipses.

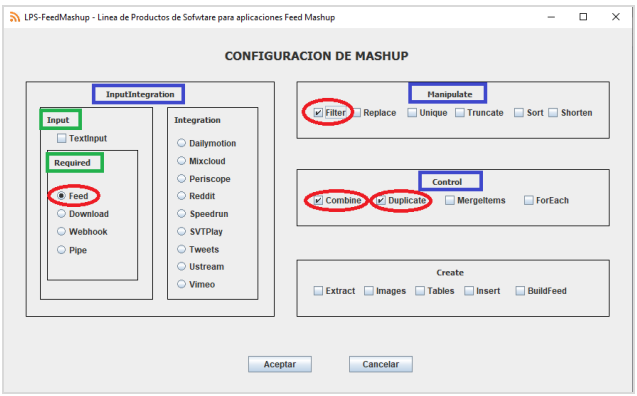

Fig. 5. Vista de la configuración de Mashup del caso de estudio 1.

Para visualizar el código generado, se debe tener instalado una versión de XAMPP (servidor web local multiplataforma que permite la creación y prueba de páginas web). En la Fig. 6, se puede observar la página generada a través de la herramienta para el caso de estudio 1, donde se introduce desde la barra de direcciones del navegador el siguiente texto: http://localhost/casoestudio.html.

La ejecución del caso de estudio a través de la herramienta, comprobó que funciona correctamente, ya que fue posible seleccionar diversas configuraciones que incluyen las características del MC. Tanto el código generado por la herramienta como su ejecución es correcta, como así fue posible generar el código fuente automáticamente como visualizar los feeds RSS creados.

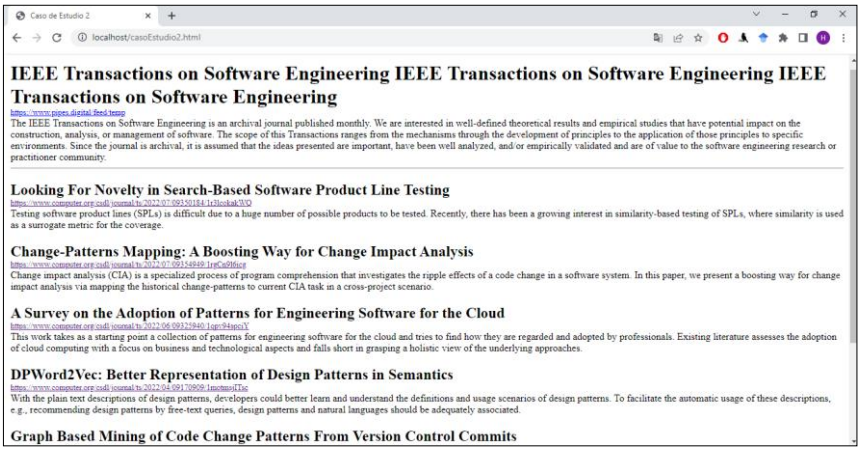

Fig. 6. Vista de la página HTML generada para el caso de estudio 1.

6.2 Evaluación de LPS-FeedMashup

Se realizó una evaluación de la calidad de la LPS utilizando el método propuesto en [32]. Este método se basa en el estándar de calidad ISO/IEC 25000 SQuaRE, su principal ventaja es que se puede aplicar a cualquier LPS desarrollada.

En este estudio, se aplicó este método para evaluar la mantenibilidad de la LPS. La mantenibilidad es el grado con el cual el producto software puede ser modificado. Un alto grado de mantenibilidad indica que un producto está estructurado de una manera que lo hace fácil de modificar. La comunalidad se evalúa teniendo en cuenta aspectos relacionados con la reutilización de los activos debido a que son partes comunes de diferentes productos, y la variabilidad se evalúa en función de la riqueza y complejidad de la LPS. Los atributos y métricas para la comunalidad implican la comunalidad funcional, la cobertura funcional de un activo, y la aplicabilidad de un activo. Los atributos y métricas para la variabilidad, incluyen el cubrimiento de la variabilidad en un activo, y la complejidad de la variabilidad de la LPS, la cual se calcula de acuerdo al número de variabilidades en la LPS.

Los resultados de la evaluación se presentan en la Tabla 4. Algunos datos se extraen desde la perspectiva FeatureIDE; y otros se contabilizan desde la representación gráfica del MC presentado en la Sección 4.

El MC definido para la creación de la LPS propuesta, presenta un grado de mantenibilidad aceptable. El nivel de reutilización de un activo cubierto por las características funcionales del modelo no alcanza el valor medio posible, mientras que la cobertura de la variabilidad es favorable. El predominio de características opcionales y alternativas (variabilidad) proporciona al modelo de un alto nivel de flexibilidad.

Tabla 4. Resultado evaluación calidad de la LPS.

Atributo: Modularidad de la LPS Métrica: Ratio de activos por características (X) Formula X = A / B (A = número de activos que desarrollan en la LPS, B = número de características en el MC) Valores aceptados: X > 0.7 Resultado: X = 0.73 Conclusión: El resultado es mayor a 0.7 por lo cual la modularidad es aceptable
Atributo: Cubrimiento de la variabilidad en un activo Métrica: Cubrimiento de la variabilidad en un activo (CV) Formula: CV = A / B (A = número de puntos implementados en el activo, B = número de puntos de variación incluidos en el alcance de la LPS) Valores aceptados: CV < 0.3 Resultado: CV = 0.27 Conclusión: El valor CV es aceptado, por lo tanto, crece la reutilización
Atributo: Comunalidad funcional Métrica: Cubrimiento funcional de un activo (FC) Fórmula: FC = S / N (S = sumatoria de A / B para cada i, tal que i = 1 hasta N, donde A = Número de aplicaciones utilizando la característica funcional i, B = Número total de aplicaciones en la LPS, N = Número total de características funcionales) Valores aceptados: FC > 0.7 Resultado: FC = 0.45 Conclusión: El valor FC no es superior a 0.7, no se alcanzó una comunalidad funcional aceptable, debido a que las funcionalidades son opcionales o alternativos.
Atributo: Aplicabilidad de un activo Métrica: Aplicabilidad acumulativa (CA) Formula: CA = 0.5 * FC + 0.5 * CV (FC = Cubrimiento funcional de un activo, CV = Cubrimiento de la variabilidad en un activo) Valores aceptados: CA > 0.5 Resultado: CA = 0.36 Conclusión: El resultado de CA se vio directamente afectado por el valor de comunalidad funcional, obteniendo un indicador no aceptable.
Atributo: Complejidad de la variabilidad Métrica: Puntos de variación en un caso de uso (V) Formula: V = (A + B) / N (A = número de características alternativas, B = número de características opcionales, N = número total de características) Valores aceptados: V > 0.5 Resultado: V = 0.85 Conclusión: La mayoría de las características (concretas) son opcionales o alternativas, por lo que la variabilidad es alta.

7 Conclusiones

El presente artículo procura ofrecer soluciones a los problemas relacionados al desarrollo de las aplicaciones Feed Mashup; se identifican estrategias de solución que permiten mejorar la integración y composición de estas aplicaciones.

La estrategia propuesta consiste en un modelo de características genérico que permite definir y representar una LPS para la derivación de aplicaciones en este dominio. Como resultado, se realiza su implementación a través de una herramienta denominada LPS-FeedMashup[1], que automatiza la generación de productos, en codificación Javascript, obteniendo artefactos para mejorar la reusabilidad, variabilidad y configuración necesarios para la integración y composición de estas aplicaciones, en un menor tiempo y esfuerzo.

Las evaluaciones realizadas indican que los productos generados con este enfoque están en un nivel similar a las aplicaciones que se produjeron manualmente, además el grado de mantenibilidad demostrado permitirá que la incorporación de nuevas funcionalidades al modelo, sea fácil de extender, modificar y/o corregir en la LPS. El enfoque propuesto promueve la generación automática de productos.

A diferencia de lo propuesto en los distintos trabajos relacionados analizados, los cuales se enfocan en distintas técnicas de composición (descubrimiento y recomendaciones) y usabilidad, para el desarrollo de mashup; este trabajo utiliza un enfoque de reusabilidad y variabilidad aplicado al dominio de los feeds.

Este trabajo es relevante dado que apunta a contribuir en dos campos: desarrollo web mashup, y desarrollo de software orientado a características, como así también, en dos aspectos esenciales de la Ingeniería de Software: la reutilización del software y el mantenimiento del mismo. Estos factores inciden directamente en los costos, esfuerzos y duración requeridos en el desarrollo del software.

Los objetivos planteados como trabajo futuro son, (1) incorporar otras funcionalidades u operaciones al modelo, que permita la evolución del MC, (2) analizar e implementar la creación de la herramienta en entorno Web, obteniéndose una mayor proyección en su aplicación en línea, (3) agregar un módulo a la herramienta que permitirá seleccionar diferentes lenguajes como salida para el código generado, y (4) optimizar el código generado por la herramienta.

Referencias

1. Daniel, F., Muhammand, I., Soi, S., De Angeli, A., Wikinson, C., Casati, F., Marchese, M.: Developing Mashup Tools for End.Users: On the Importance of the Application Domain, International Journal on Next-generation Computing, Vol. 2, Nro. 2 (2012).
2. Daniel, F., Matera, M.: Mashups Concepts, Models and Architectures. Milano, Italy: ISBN: 978-3-642-55049-2, Springer (2014).

[1] Código fuente disponible en GitHub: https://github.com/gispunpauarg/LPS-FeedMashup

3. Singh, G., Sahu, S.: Review on "Really Simple Syndication (RSS) Technology Tools". In 2015 IEEE International Conference on Computational Intelligence & Communication Technology (pp. 757-761), IEEE (2015).
4. Mahboob, K., Ali, F., Nizami, H.: Sentiment analysis of RSS feeds on sports news–a case study. International Journal of Information Technology and Computer Science, 11(12), 19-29 (2019).
5. Tinajero Diaz, I.: Composición de sistemas con Mashups: El caso PhysicallTrello. Centro de Investigación de Matemática A.C. Zacatecas (2016).
6. Clements, P., Northrop, L.: Software product lines. Addison-Wesley (2002).
7. Pohl, K., Boeckle, G., Van Der Linden, F.: Software Product Line Engineering-Foundations, Principles, and Techniques. Springer Verlag, Berlin/Heidelberg (2005).
8. Acher, M., Baudry, B., Heymans, P., Cleve, A., Hainaut, J. L.: Support for reverse engineering and maintaining feature models. In Proceedings of the 7th International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS '13), Association for Computing Machinery, New York, NY, USA, Article 20, 1–8. https://doi.org/10.1145/2430502.2430530 (2013).
9. Galindo, J., Alférez, M., Acher, M., Baudry, B., Benavides, D.: A variability-based testing approach for synthesizing video sequences. In Proceedings of the 2014 International Symposium on Software Testing and Analysis, pp. 293-303 (2014).
10. Jiang, B., Liu, P., Wang, Y., Chen, Y.: HyOASAM: A hybrid open API selection approach for mashup development. Mathematical Problems in Engineering, vol. 2020, Article ID 4984375, 16 pages. https://doi.org/10.1155/2020/4984375 (2020).
11. Gu, Q., Cao, J., Peng, Q.: Service package recommendation for mashup creation via mashup textual description mining. In 2016 IEEE International Conference on Web Services (ICWS) (pp. 452-459). IEEE (2016).
12. Paredes-Valverde, M. A., Alor-Hernández, G., Rodríguez-González, A., Valencia-García, R., Jiménez-Domingo, E.: A systematic review of tools, languages, and methodologies for mashup development. Software: Practice and Experience, 45(3), 365-397 (2015).
13. Li, C., Zhang, R., Huai, J., Sun, H.: A novel approach for API recommendation in mashup development. In 2014 IEEE International Conference on Web Services (pp. 289-296). IEEE (2014).
14. Albinola, M., Baresi, L., Carcano, M., Guinea, S.: Mashlight: a lightweight mashup framework for everyone. In 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (2009).
15. Zhao, Q., Huang, G., Huang, J., Liu, X., Mei, H.: A web-based mashup environment for on-the-fly service composition. In 2008 IEEE International Symposium on Service-Oriented System Engineering (pp. 32-37). (2008).
16. Huang, A. F., Huang, S. B., Lee, E. Y., Yang, S. J.: Improving end-user programming with situational mashups in web 2.0 environment. In 2008 IEEE International Symposium on Service-Oriented System Engineering (pp. 62-67). (2008).
17. Lian, S., Tang, M.: API recommendation for Mashup creation based on neural graph collaborative filtering, Connection Science, 34:1, 124-138, DOI: 10.1080/09540091.2021.1974819 (2022).
18. Northrop, L., Clements, C.: A Framework for Software Product Line Practice, Version 5.0. Software Engineering Institute-Carnegie Mellon University. Recuperado de https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=495357 (2012).
19. Capilla, R., Bosch, J., Kang, K. C.: Systems and Software Variability Management. Springer-Verlag Berlin Heidelberg. DOI:10.1007/978-3-642-36583-6 (2013).

20. Apel, S., Batory, D. S., Kästner, Ch., Saake, G.: Feature-Oriented Software Product Lines-Concepts and Implementation. Springer. https: //doi.org/10.1007/978-3-642-37521-7 (2013)
21. Kang, K., Cohen, S., Hess J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI90-TR-21, Software Engineering Institute, Carnegie Mellon University, November (1990).
22. Schobbens, P., Heymans, P., Trigaux, J., Bontemps, Y.: Generic semantics of feature diagrams. Comput. Netw., Vol. 51, Nro. 2, pp. 456-479 (2007).
23. Horcas Aguilera, J., Pinto, M., Fuentes, L.: Software Product Line Engineering: A Practical Experience. In SPLC 2019: 23rd International Systems and Software Product Line Conference (164-176), Paris, France: ACM Digital Library (2019).
24. Thüm, T., Kästner, C., Benduhn, F., Meinicke, J., Saake, G., Leich, T.: FeatureIDE: An extensible framework for feature-oriented software development. Science of Computer Programming, 70–85. https://doi.org/10.1016/j.scico.2012.06.002 (2014).
25. González, A., Luna, C., Zorzan F., Szasz N.: Automatic Derivation of Behavior of Products in a Software Product Line. IEEE Latin America Transactions, Vol. 12, Nro. 6 (2014).
26. Gherardi, L. Brugali, D.: An eclipse-based Feature Models toolchain. In Eclipse-IT 2011. The Sixth Workshop of the Italian Eclipse Community, pp. 242-253 (2011).
27. Oyarzo, F., Herrera, F., Miranda, M., Casas, S.: Línea de Productos de Software para aplicaciones de TVDi asado en patrones de diseño. ISSN 1852 - 4516. https://publicaciones.unpa.edu.ar/index.php/ICTUNPA/article/view/503 (2013).
28. Vázquez-Ingelmo, A., Therón, R.: Beneficios de la aplicación del paradigma de líneas de productos software para generar dashboards en contextos educativos. RIED. Revista Iberoamericana de Educación a Distancia, 23(2), 169-185 (2020).
29. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of MIS (24:3), 45-77 (2008).
30. Offermann, P., Levina, O., Schönherr, M., y Bub, U.: Outline of a Design Science Research Process. DESRIST '09, ACM, New York (2009),
31. Apel, S., Kästner, C., Liebig, J.: FeatureHouse Project. [online]. FeatureHouse: https://www.se.cs.uni-saarland.de/apel/fh/. (2011).
32. Montagud, G. S.: Un Método para la Evaluación de la Calidad de Líneas de Productos Software basado en SQuaRE. Grupo de Ingeniería de Software y Sistemas de Información (ISSI) (2009).