UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Maestría en Gestión Estratégica de Tecnología de la Información

Segunda Cohorte

SML: Un lenguaje de dominio específico para monitoreo de sistemas

---

Trabajo de titulación previo a la obtención del título de Magíster en Gestión Estratégica de Tecnologías de la Información

---

Autor:

Ing. Edgar Andrés García Clavijo
CI: 0104638507
Correo electrónico: andres.garcia.clavijo@hotmail.com

Directora:

Ing. Irene Priscila Cedillo Orellana PhD.
CI: 0102815842

Cuenca, Ecuador

18-junio-2021

Resumen

Las buenas prácticas de desarrollo de software sugieren que los sistemas incluyan funcionalidades de monitoreo, permitiendo la verificación, auditoría, trazabilidad de las operaciones y rápida respuesta frente a incidentes. Por otro lado, los lenguajes de modelado de dominio específico (DSML) muestran gran utilidad al permitir plasmar el conocimiento a un alto nivel de abstracción. Incluso, la aplicación de herramientas de transformación de DSML hacia implementaciones específicas reduce considerablemente el esfuerzo y tiempo de desarrollo, optimizando los recursos en el diseño de los sistemas, sin preocuparse de detalles de bajo nivel.

En este proyecto se propone un lenguaje de modelado de dominio específico orientado a la monitorización de sistemas de software, con enfoque a la generación de plataformas en la nube para servicios de monitoreo. Para esto, se recopilaron las necesidades de monitoreo de diferentes áreas y aplicaciones, se sintetizaron los conceptos relevantes, se consolidaron necesidades en común permitiendo generalizar y dotar al lenguaje de la suficiente semántica y expresividad para la mayoría de dominios. Se dotó al lenguaje de modelado de sintaxis y semántica, definiendo los componentes gráficos que permiten expresar diferentes necesidades de monitoreo, y especificar restricciones, contextos y notaciones, respectivamente.

Finalmente, cabe destacar, que este estudio busca contribuir con la industria 4.0, proporcionando una herramienta de diseño para facilitar el desarrollo de soluciones de monitoreo de sistemas. El lenguaje para monitoreo de sistemas, por sus siglas en inglés (SML) está orientado a la integración de datos en sistemas de sistemas, extendiendo el alcance del monitoreo hacia la perspectiva de un tercero (entidad de regulación, auditoria o servicios de monitoreo en la nube).

Palabras claves: DSML. Lenguaje de modelado. Monitoreo de sistemas.

Abstract

Best practices in software development suggest that systems include monitoring functionalities, allowing verification, auditing, traceability of operations, and quick response to incidents. On the other hand, domain-specific modeling languages (DSML) have shown great utility by allowing them to portray knowledge at a high level of abstraction. Even the application of DSML transformation tools towards specific implementations considerably reduces development time and effort, optimizing resources to take advantage of them in improving the design of systems, without worrying about low-level details.

In this work, a domain-specific modeling language oriented to systems monitoring specification is proposed, with a focus on the generation of cloud platforms for monitoring services. For this, the monitoring needs of different areas and applications were considered, the relevant concepts were synthesized, everyday needs were unified and consolidated, allowing to generalize and provide the modeling language with sufficient expression capabilities for monitoring the most domains. The syntax and semantics modeling language was provided, defining the graphical components that allow expressing each of the monitoring needs, and specifying restrictions, contexts, and notations, respectively.

This study aims to contribute to Industry 4.0, with a design tool to facilitate the development processes of system monitoring solutions. This language is oriented to the integration of data in system of systems, extending the scope of the monitoring towards the perspective of a third party (regulatory entity, audit or monitoring services in the cloud).

Keywords: DSML. Modeling language. Systems monitoring.

Índice de contenido

Índice de figuras

Índice de tablas

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Edgar Andrés García Clavijo en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación SML: Un lenguaje de dominio específico para monitoreo de sistemas, de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 18 de Junio de 2021

Ing. Edgar Andrés García Clavijo
C.I: 0104638507

Cláusula de Propiedad Intelectual

Cláusula de Propiedad Intelectual

Yo, Edgar Andrés García Clavijo, autor del trabajo de titulación SML: Un lenguaje de dominio específico para monitoreo de sistemas, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 18 de Junio de 2021

Ing. Edgar Andrés García Clavijo
C.I: 0104638507

Agradecimientos

Este trabajo de titulación ha sido posible en base al trabajo y esfuerzo de quienes con mucha dedicación han permitido alcanzar los objetivos planteados. En este sentido, agradezco a la Ing. Priscila Cedillo Orellana PhD. por su apoyo y dedicación al dirigir e impartir su conocimiento para desarrollar este proyecto, por estar siempre pendiente y dispuesta a ayudar. De igual manera, agradezco a los docentes de la Maestría en Gestión Estratégica de Tecnologías de la Información, de quienes he recibido formación académica que ha resultado de gran utilidad en este trabajo. Finalmente, expreso mi agradecimiento al equipo de Copicomp Cia. Ltda. por ser partícipes de la validación de este trabajo en la aplicación del caso de estudio.

Andrés

Dedicatoria

Dedico este trabajo de titulación a las personas que me han apoyado en todo momento y han sido mi guía, mi fuerza y mi ejemplo. A mi abuelito Ricardo, quien siempre estuvo apoyando mis estudios. A mis padres Edgar y Miriam, que han sido mi mejor ejemplo y me han impulsado en todo momento. A mi hermano Juda, quien ha sido mi brazo derecho y mi apoyo constante. A Angélica por su todo su cariño y paciencia.

Andrés

Capítulo 1 – Introducción

En este capítulo se presentan los antecedentes, problemática actual y la solución propuesta en el presente trabajo de titulación, incluyendo la hipótesis, objetivos generales y específicos, junto con la metodología de investigación a utilizar. Finalmente, se presenta el aporte científico asociado a este proyecto de titulación.

Antecedentes

A lo largo del tiempo se han creado muchos lenguajes de programación, de acuerdo al avance tecnológico y las necesidades de la industria del software. La migración del lenguaje de programación Ensamblador hacia los lenguajes de programación de propósito general o específico, mostró altos incrementos de productividad, pues una sola línea de código en C++, Java o Visual Basic, correspondía a muchas líneas de código en lenguaje Ensamblador (Shaw, 1990).

En el desarrollo del software se ha mostrado aumento de la productividad al escoger el lenguaje de programación en un nivel de abstracción apropiado. Por ejemplo, Diprose et al. (2016) proponen una API de programación para aplicaciones de robots en interacciones sociales, la cual muestra sus ventajas al tener un nivel alto de abstracción. De la misma forma, el desarrollo dirigido por modelos está enfocado en incrementar la productividad y reducir el *time-to-market* al crear cercanía con el dominio del problema junto con herramientas de transformación a código (Sendall & Kozaczynsk, 2003).

En este sentido, entran en juego los lenguajes de dominio específico (DSL) cuyo principio es diseñar y desarrollar sistemas desde un nivel más alto de abstracción (de forma gráfica) al representar varias facetas del sistema en cuestión, contextualizado en un dominio específico, permitiendo reducir los esfuerzos de desarrollo sin entrar en detalles de bajo nivel, siguiendo las pautas generales para satisfacer las necesidades del usuario (Paige, Ostroff, & Brooke, 2000) .

En el trabajo de Wortmann et al. (2020) se ha realizado un mapeo sistemático extendido de literatura sobre lenguajes de dominio específico en la industria del software. Los resultados muestran que la mayoría de estudios están enfocados en la representación digital de sistemas, interfaces, modelos de datos, integración y configuración. El estudio también muestra que desde el año 2018 se ha incrementado en un 450% en métodos para modelado, validación, verificación y manejo de errores en productos de software.

En general, los estudios existentes están orientados a reducir el costo en la integración de sistemas como lo menciona Garcia et al. (2016) , ahorrar energía en la reconfiguración de sistemas según lo indica Mechs et al. (2013) y mantener la competitividad según Strang & Anderl (2014). Los resultados de la revisión sistemática de literatura de Wortmann et al. (2020) que contempla los últimos 9 años, indican que el 69.71% de estudios contribuyen con métodos, el 13.52% corresponden a teoría y conceptos, el 9.61% propone herramientas, el 0.49% está enfocado a métricas y el resto de estudios corresponden a otras temáticas.

Planteamiento del problema

En un mundo “interconectado”, los sistemas y aplicaciones informáticas interactúan entre sí para proveer servicios y funcionalidades a los usuarios. Las buenas prácticas de desarrollo establecen que se deben mantener registros de las operaciones y transacciones que se realizan en dichos sistemas, con fines de auditoría, gestión y control de la información, detección de amenazas o comportamientos inadecuados, respuesta ante incidentes, continuidad del negocio y análisis para toma de decisiones.

El estado actual de la literatura muestra que existe una brecha con respecto a la aplicación de lenguajes de dominio específico para la definición gráfica de requisitos de monitoreo de sistemas informáticos.

El presente proyecto contribuye con la industria del software con respecto a la aplicación de metodologías modernas para mejorar la eficiencia de los procesos de desarrollo, reducir el esfuerzo en la definición de requisitos y el desarrollo de sistemas, mejorar el control en producción de los servicios desplegados, analizar la calidad de los servicios brindados en los sistemas, con información más precisa para la toma de decisiones en el negocio.

Solución propuesta

En el presente proyecto se propone la creación de un lenguaje gráfico de modelado para el dominio del monitoreo de sistemas. Para esto, se realiza una investigación sobre las mejores prácticas en los temas relacionados a monitoreo de sistemas, se analizan los casos genéricos de monitoreo más utilizados, se diseñan los componentes gráficos que conformarán el lenguaje de modelado, se definen interacciones entre dichos componentes, de manera que permitan expresar mediante un modelo gráfico la mayoría de escenarios o arquitecturas de monitoreo en los sistemas de software, dotando de significado a los modelos generados con éste lenguaje.

La propuesta del lenguaje de dominio específico va intrínsecamente relacionada a la transformación del modelo gráfico a código fuente, para lo cual se genera un prototipo de herramienta gráfica que permite transformar los modelos gráficos del DSML hacia código fuente. En este sentido, no se profundiza sobre el proceso, la metodología, las herramientas o tecnologías utilizadas para la implementación del prototipo.

Finalmente, se realiza un caso de estudio que permita ejemplificar los resultados de la utilización del lenguaje de modelado propuesto en este proyecto. Para la aplicación de este caso de estudio se considera una pequeña empresa de la ciudad, y se realiza la integración de sus sistemas con la plataforma web generada en base al modelado gráfico de las necesidades de monitoreo del proceso de registro de clientes en página de aterrizaje de la empresa.

En este proyecto no se profundiza ni se contemplan detalles específicos de la transformación de los modelos hacia código fuente. Todo lo correspondiente a los análisis que se puedan realizar sobre

los datos generados en el monitoreo de los sistemas, generación de indicadores, no forman parte de este proyecto.

En la figura 1 se muestra un esquema sobre el entorno de la solución propuesta. El diseñador o experto de dominio elabora un modelo gráfico con el lenguaje SML propuesto en este trabajo. Con una transformación modelo a texto se puede obtener código fuente para desplegar una plataforma de monitoreo como servicios en la nube, la cual contendrá todas las consideraciones de dominio que han sido incluidas en el modelo. Los sistemas en análisis se encuentran comunicados con la plataforma, manteniendo actualizado el estado de sus transacciones y procesos. Los datos generados en la plataforma pueden ser analizados o sujetos a auditoría u otros procesos.

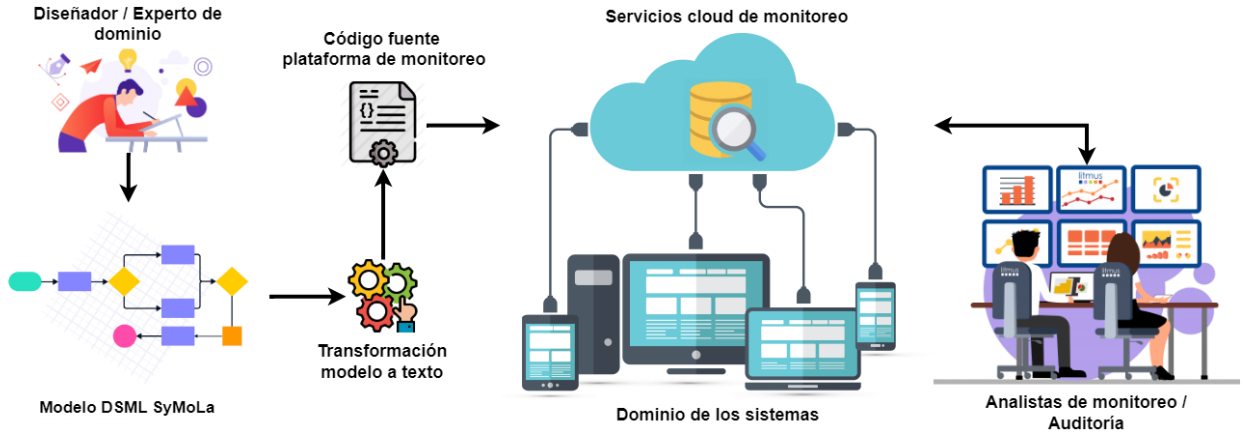

Figura 1-Esquema general de la solución

Hipótesis y Objetivos

Se definen a continuación la hipótesis y los objetivos.

Hipótesis

$H_0 =$ El lenguaje de modelado propuesto no permite especificar necesidades de monitoreo de sistemas de forma gráfica, ni reducir el esfuerzo en la implementación y soporte operativo, acorde a las necesidades establecidas en el modelo.

$H_a =$ El lenguaje de modelado propuesto permite especificar necesidades de monitoreo de sistemas de forma gráfica, reduciendo el esfuerzo en el proceso de implementación y soporte operativo, acorde a las necesidades establecidas en el modelo.

Objetivo General

Creación de un DSML orientado al monitoreo de sistemas informáticos.

Objetivos específicos

- Diseño del DSML orientado a las funcionalidades de monitoreo.

- Desarrollo de un prototipo de modelado con SML para la transformación modelo a texto, que permita utilizar el código fuente generado como base para desplegar servicios de monitoreo en la nube.
- Aplicación de un caso de estudio mediante la simulación de un escenario de monitoreo de sistemas mediante el uso del lenguaje SML, con utilización del código fuente generado para despliegue de servicios de monitoreo en la nube.

Metodología de la investigación

El proceso utilizado para desarrollar esta propuesta, está basado en una metodología estructurada conforme al modelo de transferencia tecnológica de Runeson & Höst (2009), el cual plantea ocho actividades para encontrar una solución realista por medio de un proceso iterativo de validación empírica para soluciones candidatas, como se muestra en la figura 2 y se detalla a continuación:

1. Análisis del problema: Se comprende el problema que motiva a la investigación, realizando una observación del dominio e identificando las necesidades de la industria o las empresas.

2. Planteamiento del problema: se formula el problema de forma clara y precisa, incluyendo los antecedentes del contexto del problema, objetivos que la investigación estudia, se plantean las preguntas de investigación y se realiza la justificación del estudio.

3. Revisión del estado del arte: se determina el estado del arte, analizando revisiones sistemáticas de literatura, soluciones existentes e identificando brechas que la presente investigación desea abordar.

4. Solución propuesta: se plantea una solución al problema establecido, por medio de un método definido.

5. Entrenamiento: es una actividad de tipo incremental, donde se busca proporcionar a los expertos del dominio el conocimiento necesario para tener una vista general de la solución propuesta.

6. Validación inicial: se realiza una validación inicial de la propuesta en un entorno de laboratorio, utilizando prototipos, simulaciones o estimaciones de datos, para lo cual se aplica un caso de estudio con una empresa local. No se aplica el nivel de rigurosidad de un experimento.

7. Validación realista: se realizan experimentos controlados, utilizando múltiples conjuntos de datos o casos de estudio en un entorno real de la industria.

8. Liberación de la solución: Se evalúan los resultados obtenidos, se preparan los requerimientos para liberar la solución y se despliega la solución en la industria.

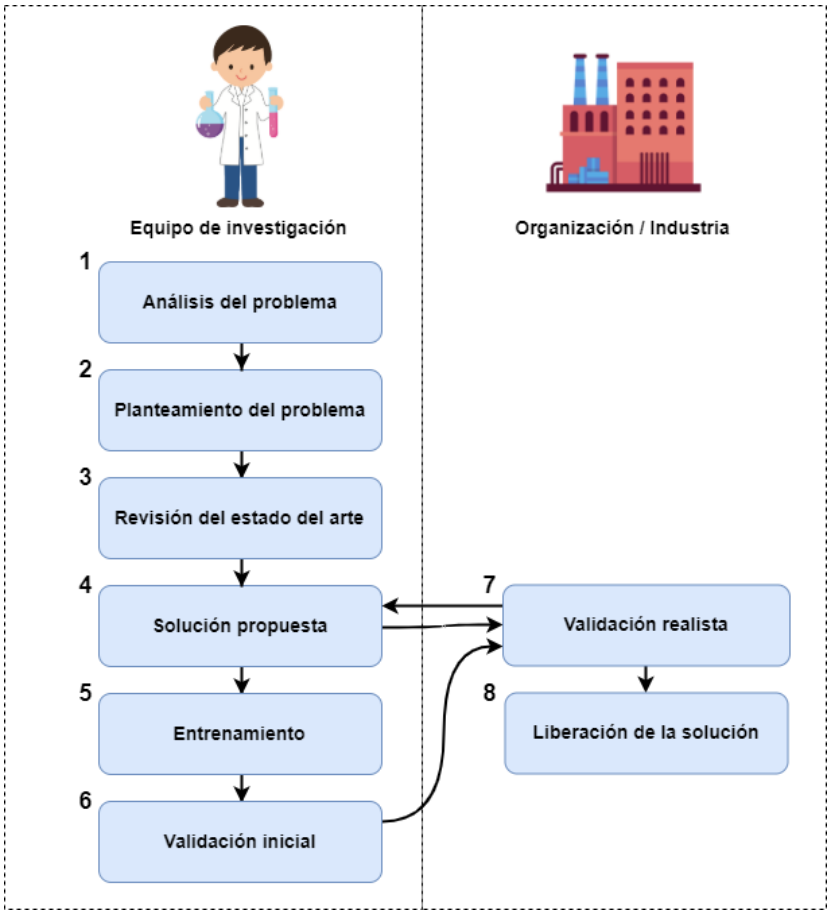

Figura 2-Proceso de metodología de la investigación

Las actividades de validación realista y liberación de la solución requieren acceso a validaciones en el sitio de trabajo, y experimentación en diferentes empresas de Ecuador. En este sentido, se ha considerado conveniente empezar con una validación local en la ciudad de Cuenca, con lo cual se esclarezca el contexto de investigación para una posterior extensión a otras ciudades. Como alcance en este proyecto, las actividades de validación realista y liberación de la solución se han considerado como trabajos futuros, abarcando únicamente hasta la fase de validación inicial.

Estructura del trabajo

La estructura de este trabajo de titulación se define de la siguiente manera:

- Capítulo 1 - Introducción: Se establecen los antecedentes, se plantea el problema proporcionando una vista general de la solución propuesta, estableciendo hipótesis y objetivos de investigación, describiendo la metodología de investigación a utilizar y mencionando artículos científicos asociados a este proyecto.
- Capítulo 2 - Marco teórico: Se han recopilado los conceptos más importantes asociados a la temática de este trabajo para proporcionar al lector un mejor entendimiento sobre

tecnologías, protocolos, herramientas y conceptos utilizados para elaborar este trabajo. También se incluyen resultados que han sido encontrados en revisiones de literatura para esclarecer el estado del arte en lo relacionado a lenguajes de dominio específico.

- Capítulo 3 - SML: Se describe la metodología que se ha utilizado para desarrollar el lenguaje de modelado para monitoreo de sistemas SML. Se definen fases y se describe a detalle el lenguaje propuesto en los aspectos de sintaxis, semántica y restricciones. Adicionalmente, se incluye una sección que muestra un enfoque de servicios en la nube como proyección del alcance de este trabajo.
- Capítulo 4 – Validación: Se establece un caso de uso para la validación inicial de una implementación específica del lenguaje SML en el proceso de diseño de una solución de monitoreo con servicios en la nube, se incluyen resultados y discusión.
- Capítulo 5 - Conclusiones y trabajo futuro: Se presenta la información importante obtenida con este proyecto de tesis y se proponen trabajos futuros en continuidad de esta investigación.

Aporte científico

Este proyecto de titulación tiene asociado un documento científico, presentado en una conferencia internacional:

"Towards a System Monitoring Modeling Language (SyMoLa)" presentado en la International Conference of Digital Transformation and Innovation - INCODTRIN 2020 con sede en Quito, Ecuador llevada a cabo del 28 al 30 de octubre de 2020 (García & Cedillo, 2020).

Capítulo 2 - Marco teórico

En este capítulo se definen diferentes conceptos que son abordados a lo largo de este trabajo de titulación, como base para un correcto entendimiento del mismo. A continuación, se presentan tecnologías y elementos relacionados a este trabajo de titulación.

Conceptos

Monitoreo de sistemas web

Lauriac (2016) establece que el monitoreo es el proceso que permite recolectar, tratar, analizar y difundir información, manteniendo un conjunto de actores involucrados en un proyecto o sistema establecido, permitiendo obtener información importante para la toma de decisiones.

Dotcom-Monitor (2020) propone que el monitoreo puede ser utilizado para cubrir gran variedad de propósitos, por ejemplo:

- Supervisar transacciones web, generando trazabilidad del estado y de todo lo que interviene entre las partes que realizan la transacción.
- Supervisar la disponibilidad de aplicaciones, garantizando una detección oportuna de anomalías, caídas de servicio u otros inconvenientes que pudieran ocurrir.
- Monitoreo de la página de aterrizaje, permitiendo conocer estadísticas de acceso al sitio web y preferencias de interacción de los usuarios con la página.
- Monitoreo de envío de formularios, para verificar que el procesamiento de los formularios en el servidor se realiza correctamente y para obtener estadísticas sobre el tiempo de respuesta.
- Monitoreo del carrito de compras, para analizar tendencias en el consumo de los clientes y planificar el abastecimiento de productos.

DSML

Los lenguajes de modelado de dominio específico, por sus siglas en inglés (DSML) tienen como objetivo elevar el nivel de abstracción más allá de la programación, especificando la solución en un lenguaje que utiliza directamente conceptos y reglas de dominio de un problema específico. De la misma forma, busca generar productos finales en un lenguaje de programación elegido u otra especificación de alto nivel. La automatización del desarrollo de aplicaciones puede extender su aplicabilidad, pues el lenguaje de modelado, el generador de código y el código generado se ajustan a los requisitos de un dominio de aplicación específico. En otras palabras, son específicos del dominio y están completamente bajo el control de sus usuarios expertos en el área de estudio (Steven & Tolvanen, 2008).

Ingeniería basada en modelos

La ingeniería dirigida por modelos (MDE) está basada en el modelado como actividad principal en el ciclo de vida de un proyecto de software. Se fundamenta en los modelos como artefactos principales

para el desarrollo del software, antes que en el código fuente (Cuadrado, Cánovas Izquierdo, & Molina, 2014).

Rodrigues Da Silva (2015) menciona que el objetivo de este paradigma de desarrollo es aumentar la productividad al aumentar la compatibilidad entre los sistemas reutilizando los modelos estandarizados. También busca simplificar el proceso de diseño, aumentando la comunicación entre el equipo que trabaja en el sistema.

La combinación de las técnicas de ingeniería, junto con los lenguajes de modelado de dominio específico ofrece gran aplicabilidad en la forma en que se desarrollan las aplicaciones. Trabajar en un nivel de abstracción más alto y especializarse en desarrollo de un dominio particular, tiene potencial para convertir a expertos de dominio en desarrolladores de aplicaciones (Sousa, Costa, Clarke, & Allen, 2012).

UML

El lenguaje de modelado unificado (UML) pretende ser un lenguaje universal para el modelado de sistemas, permitiendo expresar conocimiento de diferentes tipos o propósitos. La independencia que mantiene UML entre el modelado y el proceso para generar una solución, genera libertad para expresar el conocimiento de un dominio específico, de forma estándar en la industria (Booch & Rumbaugh, 1996).

Cloud computing

La computación en la nube comprende un nuevo paradigma para almacenar y proveer servicios en el Internet. Tiene múltiples ventajas para las empresas y para los clientes de los servicios, pues evita la planificación previa de aprovisionamiento de los servicios, permitiendo escalar en base a la demanda, reduce los costos para las empresas que ingresar al comercio electrónico, permite acceso inmediato a recursos de hardware de forma ubicua sin necesidad de inversiones, reduce las barreras a la innovación en tecnologías de la información, aumenta el alcance de los negocios a un contexto virtual (Olaru, 2014).

Servicios web

Los servicios web son componentes de software diseñados para proveer de interoperabilidad entre sistemas con plataformas, fuentes de datos o ubicaciones heterogéneas. Permiten enlazar capas de una aplicación tradicional, proveer servicios genéricos para una organización o realizar integración entre los sistemas de las organizaciones. El principio básico de su funcionamiento es el paso de mensajes, para esto se utilizan tecnologías asociadas como SOAP, REST, HTTP, XML, entre otras (Mockford, 2004).

Java

Java es un lenguaje de programación de propósito general, basado en clases, concurrente y orientado a objetos. Su diseño busca la simplicidad, manteniendo similitudes con otros lenguajes como C o C++. Es fuertemente tipado, es decir, los tipos de datos que utiliza deben ser declarados

explícitamente. El código fuente Java es compilado a código binario, el cual es ejecutado en la máquina virtual de Java (Gosling & Buckley, 2011).

SOAP

De acuerdo a Nielsen & Thatte (2014) el protocolo de acceso de objeto simple, por sus siglas en inglés (SOAP) es un protocolo para intercambio de información en un entorno descentralizado y distribuido. Está basado en XML y consiste en tres partes: una envoltura que define lo que contiene un mensaje y como procesarlo, un conjunto de reglas de codificación para tipos de datos, y una convención para representar llamadas y respuestas remotas. Los servicios web que utilizan SOAP en su mensajería pueden ser descritos conforme un estándar denominado especificación WSDL, la cual contiene todas las características inherentes al servicio web que es desplegado para exponer sus servicios de la manera prescrita por su descripción, normalmente transmitidos mediante HTTP con una serialización XML junto con otros estándares relacionados con la Web (Mockford, 2004).

REST

El protocolo de transferencia de estado representacional, por sus siglas en inglés (REST) es un estilo arquitectural derivado de la web, en el cual prevalecen los principios que permiten gran escalabilidad, crecimiento y éxito en Internet. Los servicios web RESTful utilizan las características existentes del protocolo HTTP, manteniendo la comunicación en torno a transferencia de representaciones de los recursos. Con respecto a SOAP, REST ofrece mayor simplicidad, flexibilidad en sus interfaces, interoperabilidad y escalabilidad. En el rendimiento de las aplicaciones o servicios web se nota gran diferencia a favor de REST, por lo cual está siendo altamente utilizado (August, No, & Rathod, 2017).

Socket

Un Socket es un tipo de estructura abstracta de datos, provista por un sistema operativo para construir un acceso que permita el envío o recepción de mensajes. Las comunicaciones TCP con Sockets operan en la capa de transporte del modelo TCP/IP, manteniendo una comunicación orientada a la conexión para la completitud e integridad de los mensajes a transmitir. Los mensajes son transmitidos como una cadena de bytes en un segmento de mensaje, se incluyen controles de tiempo, confirmación de recepción. Muchas aplicaciones conocidas como Telnet, FTP, SMTP utilizan comunicaciones TCP (Zhu, 2009).

OBEO Designer

OBEO *Designer* es una herramienta de modelado adaptable basada en puntos de vista. Dispone de un entorno de configuraciones para diferentes representaciones. Está basado en las tecnologías del entorno de trabajo de Eclipse, junto con Entity Modeling Framework para proveer los elementos necesarios para la elaboración de modelos. Esta herramienta permite a los arquitectos crear herramientas de modelado gráfico que soporten su propio lenguaje, notación, procesos y objetivos técnicos. El modelado puede ser realizado hacia representaciones en diagramas, tablas o árboles (Kouhen et al., 2012).

Los diseños con OBEO *Designer* obedecen a tres pasos principales:

1. Definición del vocabulario de dominio
   En esta etapa se definen conceptos de dominio, relaciones y propiedades al crear un metamodelo bajo el esquema *ecore*. Para esto se utiliza un editor gráfico de OBEO y se obtiene como producto generado una implementación del metamodelo.
2. Creación y visualización de modelo semántico
   Luego de haber realizado la descripción del editor, se crea un modelo semántico y se visualiza representaciones gráficas en forma de diagramas, tablas o árboles.
3. Descripción de las representaciones gráficas
   Esta etapa consiste en definir la sintaxis concreta mediante figuras, colores, tamaños y otras propiedades, generando un enlace con el modelo semántico del paso anterior. Se incluyen validaciones, se definen paletas de herramientas o capas de filtrado, e incluso se personalizan generadores de código utilizando Acceleo.

Estado del arte

En esta sección se muestra lo más relevante con respecto al estado del arte en lenguajes de dominio específico y la aplicabilidad al monitoreo de sistemas, encontrando características y tendencias en la utilización de los mismos.

El trabajo de Mernik (2017) muestra un estudio de mapeo sistemático de literatura sobre lenguajes de dominio específico, para identificar tendencias e ideas que no han sido investigadas. En dicha investigación se muestra que los diseñadores o programadores que utilizan lenguajes de dominio específico, requieren herramientas que les permitan encontrar errores en sus desarrollos. Sin embargo, los resultados del mapeo sistemático evidencian que existen pocos estudios que analizan herramientas de depuración, pruebas automatizadas o refactorización de código. También muestra que existe una brecha de investigación en métodos formales para análisis de dominio y descripción semántica de lenguajes de dominio específico.

Por otro lado, algunos estudios muestran la tendencia en la industria del software en utilizar lenguajes de programación de alto nivel para el desarrollo de soluciones en dominios específicos. Existen múltiples lenguajes de programación para diferentes propósitos o dominios. Davison, Hines, & Muller (2009) analizan las tendencias en el desarrollo de interfaces de simulación programables para aplicaciones de neurociencia. Muestran la importancia de la expresividad, la facilidad de uso y el alcance de las interfaces para la simulación, buscando transformar las ideas en un entorno de simulación funcional. Al utilizar estos simuladores, los científicos pueden expresar modelos en términos de conceptos biológicos sin preocuparse por los detalles computacionales de la implementación de bajo nivel o código fuente. En el trabajo de Cedillo et al. (2016) se presenta una infraestructura para monitorear el cumplimiento de los acuerdos de nivel de servicio (SLA). Los autores utilizan modelos, describen componentes, artefactos y las interacciones entre ellos. La infraestructura recupera información de servicios en la nube para obtener métricas de monitoreo y

verificar el nivel de cumplimiento de los SLA. Las ventajas que se encuentran con el uso de modelos para la infraestructura de monitoreo son una mayor flexibilidad y un fácil mantenimiento frente a cambios en los SLA.

La literatura existente con respecto a lenguajes de modelado en la industria del software ha sido analizada en el estudio de Wortmann et al. (2020) por medio de un mapeo completo y sistemático de la literatura. Dicho estudio hace especial énfasis en la industria del software 4.0, la cual está centrada en proporcionar métodos para resolver desafíos de integración y representación digital. Para el estudio el autor consideró 3344 publicaciones candidatas, analizadas sistemáticamente para obtener 408 publicaciones relevantes. Los resultados muestran que la mayoría de los estudios se centran en la representación digital de interfaces, sistemas, modelos de datos, integración y configuración. De la misma forma, la revisión muestra que desde el año 2018, se ha encontrado un incremento del 450% en métodos de modelado, validación, verificación y manejo de errores en productos de software. Únicamente el 13,48% de los estudios describen explícitamente las expectativas de los autores sobre el impacto de su contribución. En general los estudios analizados en la revisión sistemática de literatura de Wortmann et al. (2020), contribuyen a reducir el tiempo de desarrollo y comercialización de las soluciones de software, reducir los costos asociados al desarrollo, facilitar la integración o configuración y aumentar la sostenibilidad y competitividad internacional. Como resultado del análisis estadístico de los estudios, Garcia et al. (2016) está orientado a reducir el coste de integración de sistemas, Mechs et al. (2013) se enfoca en el ahorro de energía en la reconfiguración de sistemas, Strang & Anderl (2014) en mantener la competitividad. Los resultados de la revisión sistemática de la literatura indican que 69,71% de los estudios aportan métodos, 13,52% corresponden a teoría y conceptos, 9,61% proponen herramientas o soluciones de software, el 0,49% de estudios se enfocan en métricas y el resto de estudios corresponden a otros temas. Basado en estas cifras, se muestra en la figura 3 la distribución de estudios de acuerdo a su categoría con respecto al tipo de contribución realizada a la industria.

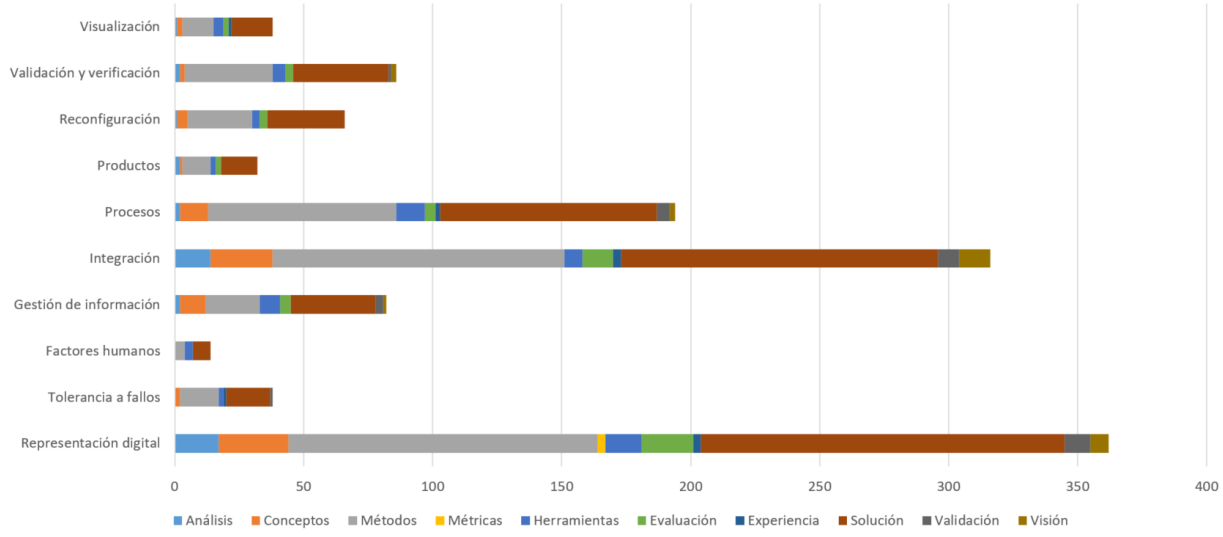

Figura 3 -Número de estudios por categoría y tipo de contribución Wortmann et al. (2020)

Desde el punto de vista de los lenguajes de modelado se han encontrado diversas categorías: técnicas de modelado de variantes del lenguaje UML, como DiSpa por Bergert, Diedrich, Germany, Kiefer, & Bär (2007), UML mecatrónico por Schubert, Gerking, & Heinzemann (2016), UMM por Mazak & Huemer (2015) y UML4IoT por Thramboulidis & Christoulakis (2016); técnicas de representación del conocimiento, como Web Ontology Language (OWL) por Harcuba & Vrba (2015) y Negri et. al (2015), Semantics Web Rules Language (SWRL) por Hildebrandt et al. (2017) y Sadigh et al. (2016); metamodelos específicos para los retos actuales de la Industria 4.0, como el metamodelo industrial para sistemas de automatización por Gutierrez & Holgado (2017) o AutomationML por Kovalenko et al. (2015); técnicas de metamodelado, como ADOxx por Efendioglu & Woitsch (2017) y Walch (2017), MetaEdit+ por Chen, Maffei, & Ferreirar (2015) y Chen et al. (2016) o Xtext por Givehchi et al. (2017); varios lenguajes de dominio específico como EXPRESS DSL para el modelado de datos de productos por Divoux, Rondeau, & Lepage (1997), el modelo de datos de fabricación virtual por Ka (2013), el lenguaje de modelado para procesos de la industria 4.0 por Petrasch & Hentschke (2016) o el entorno de modelado gráfico para procesos de producción por Lütjen & Rippel (2015). Esta información ha sido plasmada en la figura 4, que representa las líneas de aplicación en la industria 4.0 junto con las técnicas de modelado para alcanzarlas.

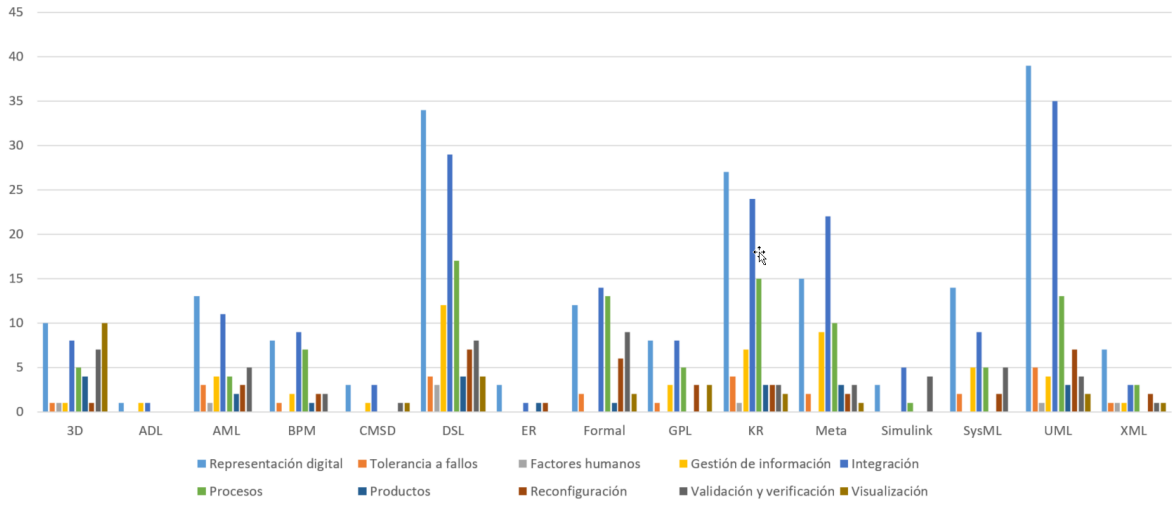

Figura 4-Líneas de aplicación en industria y técnicas de modelado Wortmann et al.(2020)

La revisión sistemática de literatura realizada por Wortmann et al. (2020), responde a una pregunta de investigación directamente relacionada a este proyecto: "¿Qué tipos de lenguajes de modelado están siendo usados en la industria 4.0 y qué áreas abordan?". El análisis de datos realizado muestra que en gran medida se utiliza UML para resolver desafíos en representación digital (39 publicaciones), integración (35 publicaciones), siendo éstas las líneas principales de contribución en los estudios analizados. También se encontró que, la verificación, validación y factores humanos, son líneas investigación que no han sido estudiadas a profundidad en el modelado de productos de la industria 4.0. Bajo este contexto, a pesar de la existencia de

numerosos estudios que proponen lenguajes de modelado para múltiples propósitos, se evidencia la brecha que existe en cuanto a lenguajes de modelado para la representación del conocimiento en el dominio de la monitorización de sistemas.

Capítulo 3 – SML

El presente capítulo describe a detalle la principal contribución de este trabajo de titulación. Se muestra la metodología utilizada para desarrollar el lenguaje *System Monitoring Language* (SML), se describe el lenguaje de modelado para monitoreo de sistemas mediante el proceso paso a paso y finalmente se propone un enfoque para servicios de monitoreo en la nube.

SML se propone como un DSML que permite modelar gráficamente el conocimiento del dominio para especificar el monitoreo de sistemas, que luego se implementaría como una plataforma en la nube. Bajo este esquema, la representación realizada en un modelo gráfico permite a los expertos del dominio especificar los procesos de sus sistemas que deben ser monitoreados, el flujo de comunicaciones entre ellos a través de mensajes, las condiciones o reglas a detectar, y los eventos que se desean llevar a cabo cuando se cumplan estas reglas.

Metodología para desarrollo del DSML

El proceso para la creación de SML se ha basado en la metodología propuesta por Chaudhuri et al. (2019). Dicha metodología ha analizado la literatura existente y en base a un conjunto de 25 pautas para definir un DSML se proponen los siguientes principios, que han sido definidos desde la perspectiva lingüística (Karsai et al. 2009).

1. Declarativo
   El lenguaje debe soportar especificaciones declarativas utilizando vocabulario de dominio para modelar el sistema. Esto permite modelar el sistema para un contexto específico utilizando las entidades, asignaciones, relaciones, comportamiento y todos los componentes declarativos propios del lenguaje que toman sentido en base al contexto utilizado.
2. Construcciones estructurales
   Para que la sintaxis del lenguaje tenga una base sólida para su utilización es fundamental que disponga de construcciones estructurales. No existe una definición específica de este concepto, sin embargo, de la misma forma que el paradigma de la programación orientada a objetos utiliza clases o métodos para soportar a sus modelos (Meyer, 2013), la sintaxis del DSML debe estar soportada por una estructura sintáctica.
3. Comportamiento
   Se deben incluir mecanismos para especificar cómo el modelo reacciona a cambios basado en su lógica interna. Por ejemplo, un diagrama de máquina de estados representa la lógica de un sistema y el detalle en sus interacciones.

Bajo este contexto la metodología propone cuatro etapas: i) capturar el conocimiento del dominio, ii) definir la máquina del dominio específico (DSM), iii) identificar y definir la estructura lingüística del DSML, iv) la sintaxis concreta.

Capturar el conocimiento del dominio

La primera etapa de la metodología consiste en capturar todo el conocimiento del dominio en estudio. Para este propósito se pueden utilizar diferentes enfoques o herramientas, como ontologías, modelado de características, mapas mentales, ingeniería orientada a modelos, entre otras. La elección de cualquier opción debe ser lo suficientemente simple para ser utilizada por expertos de dominio o clientes que no tengan un perfil técnico.

El propósito de este meta modelo es capturar conocimiento del dominio y no debe ser confundido con el meta modelo del DSML que es el producto final del proceso de esta metodología. El meta modelo de este paso permite abstraer conceptos como roles, interesados, objetivos o funciones que permiten alcanzar dichos objetivos. Adicionalmente, en este paso se define el vocabulario que será utilizado en el DSML.

Definir la máquina del domino específico

En esta etapa se utilizan patrones de arquitectura para representar el dominio en estudio. Se debe encontrar una representación que defina al modelo de dominio específico. Esta actividad la realizan los ingenieros de software con insumos de los expertos de dominio. El patrón de arquitectura seleccionado debe ser mapeado con el conocimiento de dominio capturado en el paso previo.

Identificar y definir la estructura lingüística del DSML

Todos los elementos gráficos o estructuras que el DSML debe soportar son identificados en esta etapa, basándose en el DSM obtenido en la etapa anterior. Se deben considerar los siguientes factores para identificar dichas estructuras:

1. Identificar interfaces
   Se analiza el DSM para identificar los componentes que requieren proveer información mediante la definición de una interfaz. Dicha interfaz corresponde a las funciones asignadas a cada componente, sus parámetros de entrada y salida (Karsai et al., 2009).
2. Estructuras de funciones de dominio
   Una forma de identificar las estructuras de funciones de usuario consiste en derivarlas directamente de la naturaleza de su dominio. El meta modelo del sistema es la principal fuente de conocimiento de la cual se deben identificar las funciones de dominio.
3. Modularidad
   Cuando se tienen identificadas muchas funciones de dominio que coinciden en características en común es importante poder agruparlas apropiadamente para que el lenguaje sea modular. Esto ayuda a que el lenguaje sea fácil de utilizar y pueda ser abordado

de diferentes maneras. Se pueden agrupar los componentes relacionados a una entidad en común o aquellos cuya función sea similar (Karsai et al., 2009).

4. Aspectos comunes
   Una consideración clave para la definición de los DSML es considerar aspectos que son transversales para todo el modelo de dominio. Es decir, si se observa que el DSML tiene un alcance muy grande al manejar múltiples sub-dominios en el modelo, se puede optar por agrupar en conjuntos de componentes separados para cada propósito o aspecto en común según las directrices del modelado de dominio orientado a aspectos (Gray et al. 2003).
5. Especificación de la lógica interna
   En el DSML es necesario especificar cómo las entidades reaccionan a los cambios, dotando al lenguaje de cierto dinamismo implícito. Una forma de implementar esto es alineando las construcciones estructurales con los componentes que manejen eventos o que reaccionen a cambios.

Sintaxis concreta

La última etapa de la metodología consiste en definir a grano fino los detalles correspondientes al vocabulario, las palabras clave, el uso de simbología, notaciones de infijos o sufijos para las expresiones. Para esto se debe considerar la usabilidad y la estética, y junto con el experto de dominio se debe elegir cuidadosamente cada detalle de sintaxis textual o gráfica que el DSML incluya.

Desarrollo del DSML

En esta sección se describe a detalle la contribución principal de este trabajo de titulación. Se desarrolla el proceso de la metodología para definir un DSML propuesto por Chaudhuri et al. (2019), definiendo cada etapa en la conformación del lenguaje SML.

Captura del conocimiento de dominio

Los sistemas de software actuales son muy complejos, conformados por múltiples componentes que pueden encontrarse distribuidos físicamente o virtualmente en diferentes dispositivos. El funcionamiento de cada uno de estos elementos genera información útil que permite conocer el estado actual de la operatividad del sistema, los procesos en ejecución, registros de los eventos presentados y posibles errores o incidencias. Por otra parte, cada uno de estos componentes utiliza una implementación específica o configuración personalizada para el almacenamiento de sus registros o logs de monitoreo, pudiendo ser estos en archivos de texto, bases de datos o por medio de la generación de notificaciones a los usuarios supervisores. Mientras más fuentes de información tenga el sistema a monitorear, más crecerá la complejidad del análisis a realizar en los datos, requiriendo de mucho tiempo e ingenieros con gran habilidad para dicho análisis.

En esta primera etapa se han recopilado múltiples características y consideraciones relacionadas al monitoreo de sistemas, las cuales tienen relevancia para este trabajo pues se busca que el

lenguaje de modelado para monitoreo de sistemas SML sea lo suficientemente general para expresar el conocimiento y las necesidades de monitoreo para la mayor cantidad de dominios posible.

Joyce et al. (1987) proponen un modelo de monitoreo, cuyos elementos importantes son: i) Generación: detectar eventos importantes y generar reportes de eventos y estados. Estos reportes de monitoreo sirven como insumo para construir trazas de monitoreo las cuales representan vistas históricas de la actividad del sistema; ii) Procesamiento: consolidar trazas, validar, actualizar en base de datos, combinar y encontrar correlaciones, filtrado de información, convirtiendo los datos de monitoreo de bajo nivel, al nivel de detalle y formato requerido. iii) Diseminación: enviar los reportes de monitoreo o alertas a los usuarios apropiados, administradores o agentes de procesamiento; iv) Presentación: toda la información recuperada, procesada y formateada es presentada a los usuarios en diferentes niveles de abstracción o vistas simultáneas, utilizando diferentes diagramas; v) Implementación: Sincronización de eventos para ordenamiento.

De forma similar, la solución de monitoreo propuesta por Holub et al. (2009) contempla las siguientes facetas: 1. Recolección automática de datos, 2. Normalización de datos en un formato común, 3. Análisis de correlación en tiempo de ejecución para dar una vista coherente del comportamiento del sistema, 4. Mecanismo de detección de síntomas para identificar sobre la marcha errores conocidos. Para llevar a cabo estas facetas, Holub et al. (2009) han identificado los siguientes criterios críticos: i) Sensibilidad: es de crítica importancia que la respuesta del motor de correlación en tiempo de ejecución no exceda los límites de tiempo aceptables. ii) Sobrecarga en el entorno de monitoreo: se requiere que el impacto de rendimiento en el sistema en análisis sea mínimo. iii) Escalabilidad: las aplicaciones empresariales típicas contienen gran número de componentes, por lo cual se requiere capacidad para entornos de alta carga transaccional. iv) Filtrado: es importante que el volumen de datos a ser analizado pueda ser reducido. Conviene manejar criterios como: identificación del componente, severidad, coincidencia de patrones. v) Fiabilidad: las operaciones de los sistemas empresariales durante varios días generarán grandes volúmenes de datos. Por lo cual, determinar el estado de fiabilidad de un sistema requerirá de mucho esfuerzo en el análisis de los datos. Es recomendable disponer de reportes de salud del sistema. vi) Análisis de registros sin conexión: es importante disponer de mecanismos de verificación en cualquier componente de la arquitectura, para la ejecución de pruebas unitarias. vii) Mecanismo de sincronización horaria: en redes grandes es de gran importancia disponer de un mecanismo de sincronización de reloj.

Por otro lado, Cicotti et al. (2013) proponen la herramienta de monitorización QoSMONaaS, la cual permite trabajar en un entorno basado en eventos con un sistema de comunicación de cola de mensajes como Java Message Service (JMS) para enviar consultas y obtener resultados, solicitar el servicio de monitorización, notificar violaciones de SLA, intercambio de datos y eventos relacionados con el usuario. Todas estas interacciones las implementaron mediante comunicaciones asincrónicas realizadas a través de un patrón de mensajería de publicación / suscripción en el que los canales o temas son publicados por productores de datos y suscritos por consumidores de datos.

El trabajo de Inzinger et al. (2014) presenta un marco de trabajo para la gestión eficiente en tiempo de ejecución de entornos en la nube y sistemas heterogéneos distribuidos. Propone un lenguaje de dominio específico denominado MONINA que permite definir monitoreo integrado y funcionalidades de adaptación para controlar tales sistemas. El trabajo guarda importante relación con este proyecto pues se propone una infraestructura basada en un middleware de mensajería distribuida, evitando el acoplamiento y permitiendo a los nuevos nodos a monitorear que se integren al sistema de forma dinámica. En este esquema, los expertos de negocio especifican objetivos de alto nivel para el comportamiento de las aplicaciones, las cuales son evaluadas basándose en indicadores de dominio específico sobre el estado del sistema. Las comunicaciones entre los componentes se la realiza utilizando una fábrica de mensajería distribuida que permite minimizar el tráfico innecesario en la red y que los componentes se muevan dentro de la red sin cambiar sus enlaces de conexión o perder conectividad. El lenguaje MONINA permite al usuario especificar capacidades de servicio de la plataforma, monitorear consultas y reglas de adaptación.

Los objetos principales que conforman el lenguaje MONINA consisten en: i) Eventos: Se utiliza el paradigma de interacción basada en eventos según Hutchison & Mitchell (2011) . Estos eventos son disparados desde los componentes y también desde las consultas de monitoreo para enriquecer la información de los eventos. ii) Acciones de adaptación: son invocadas por las reglas de adaptación y ejecutadas por los componentes para modificar su comportamiento. Las acciones pueden tomar parámetros. iii) Hechos: constituyen el conocimiento base para las reglas de adaptación y se derivan de los eventos de monitoreo, referenciando el tipo de evento y una clave de partición. iv) Componente: contiene toda la información necesaria para integrar sistemas de terceros en la infraestructura. Se declaran todos los eventos que emitirá, frecuencia de ocurrencia, acciones de adaptación que soporta y una referencia al host donde se encuentra desplegado el componente. v) Consultas de monitoreo: permiten el análisis, procesamiento, agregación y enriquecimiento de los eventos de monitoreo. vi) Reglas de adaptación: emplean una base de conocimiento sobre hechos sobre el estado del sistema actual, para modificar su comportamiento cuando es necesario. vii) Host: representa la infraestructura física disponible para el despliegue de los componentes, normalmente identificada por un nombre de dominio completo o una dirección IP.

Bajo este contexto, se ha consolidado lo más relevante para esta propuesta y se lo ha plasmado mediante el lenguaje de modelado unificado UML. En la figura 5 se muestra una representación del dominio de monitoreo de sistemas, al definir que un escenario de monitoreo está formado por componentes, los cuales pueden ser dispositivos IoT, servicios web, sistemas integrados o incluso procesos batch. Los componentes se comunican entre sí enviando mensajes síncronos o asíncronos. También tienen eventos y procesos, que tienen un identificador único. Por otro lado, se pueden cumplir condiciones que han sido especificadas mediante reglas, las cuales desencadenan o llaman a eventos de interés.

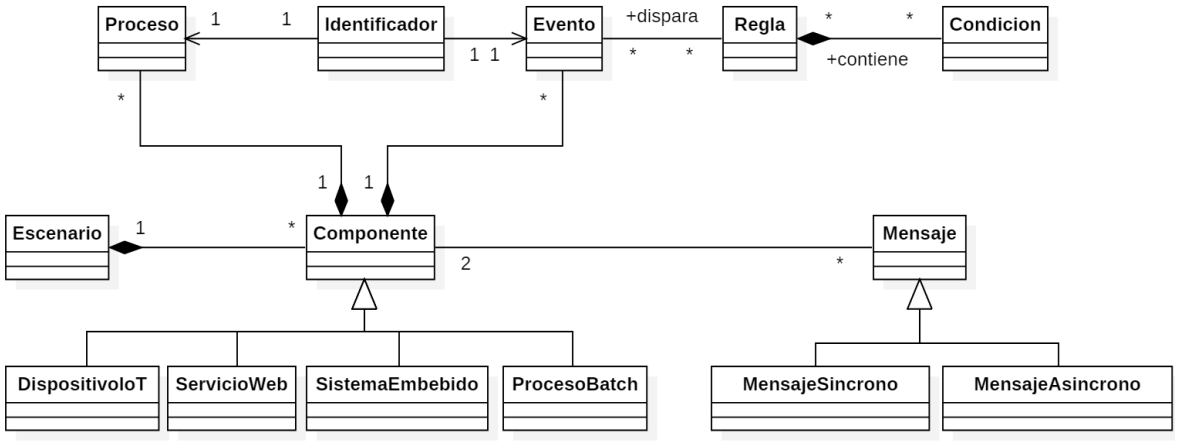

Figura 5-Representación del dominio de monitoreo de sistemas para SML

Definición de máquina de dominio específico

En esta etapa se utilizan patrones de arquitectura para definir una máquina de dominio específico (DSM). El patrón de arquitectura utilizado debe mantener relación con el conocimiento de dominio de la primera etapa. El objetivo es representar una máquina de dominio específico en términos del vocabulario de dominio. En el caso del monitoreo de sistemas, la arquitectura se puede representar como flujos de procesos para casos de éxito, o casos de error.

La sección caso de éxito se representa mediante una línea de tiempo en la cual se llevan a cabo procesos que tienen un inicio y fin definidos, y se comunican entre sí enviándose mensajes desde el fin de cada proceso hacia el inicio de otro proceso. En este caso la culminación del flujo de procesos sin interrupciones da lugar a la invocación de un evento y todo queda registrado en una base de datos. Por otro lado, la sección caso de error representa la interrupción del flujo de procesos de forma anormal. La interrupción del flujo es detectada e invoca a un evento que permite al equipo de soporte técnico revisar el incidente y actuar oportunamente. En la figura 6 se muestran estos flujos de procesos representados como máquina de dominio.

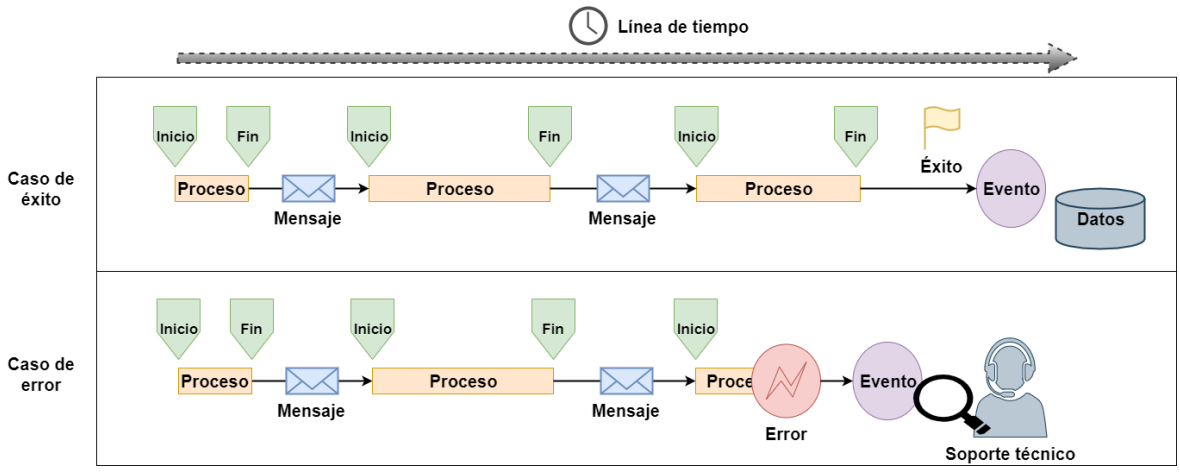

Figura 6-Representación de máquina de dominio específico para SML

Esta representación de máquina de dominio específico puede abstraer el concepto de múltiples casos de monitoreo en gran variedad de dominios. En esta representación se conserva la esencia del objetivo principal de todo mecanismo de monitoreo. El cambio de estado que sufre una transacción en su flujo de procesos es claramente definido con un inicio y fin, lo cual puede ser registrado, analizado y monitoreado para la detección de condiciones definidas en reglas que desencadenan eventos. Estos eventos puntualizan lo que se desea monitorear en el dominio dado con la información que contienen.

Identificación y definición de la estructura lingüística del DSML

En base a la etapa previa, con el DSM se deben identificar estructuras clave que soporten al DSML SML. Para esto se han analizado las interfaces que tienen los elementos del dominio, la interacción que deberían tener entre ellos, la modularidad con la que deben poder expresarse y los aspectos clave para permitir que el lenguaje de modelado pueda ser aplicado en diferentes dominios o áreas de conocimiento. Para esto se han definido diferentes elementos gráficos que conforman el lenguaje SML, los cuales se describen a continuación.

Componente:

Un **componente** representa un host de la arquitectura de sistemas a monitorear, ya sea físico o lógico. Los componentes pueden formar parte de una misma red de trabajo, encontrarse en redes diferentes o incluso en empresas diferentes, permitiendo su alcance hasta sistemas de sistemas.

La representación gráfica de un componente se muestra en la figura 7, y consiste en un recuadro con líneas entrecortadas. En la parte superior se establece el nombre del componente. En el interior del recuadro se incluyen los diferentes **procesos** que se ejecutan en dicho componente y que se requieren monitorear. En las líneas entrecortadas se pueden colocar **eventos**, los cuales son invocados tras cumplirse una determinada **regla**.

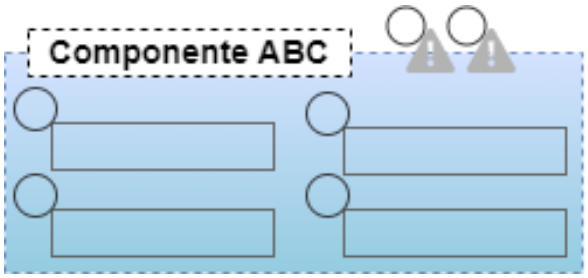

Figura 7-Componente en lenguaje SML

Proceso:

Un **proceso** representa una determinada tarea que se ejecuta en el componente del cual forma parte. Los procesos se representan gráficamente con un rectángulo, con el nombre del proceso en su interior, como se muestra en la figura 8. Cada proceso tiene un inicio y fin definidos. En la parte superior central de cada proceso se incluye un número en segundos, el cual indica la cantidad de

tiempo establecida como máxima para considerar un **timeout** durante la ejecución del correspondiente proceso.

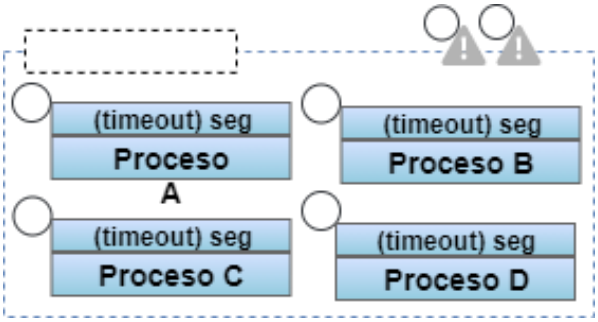

Figura 8-Proceso en lenguaje SML

Identificadores:

Un **identificador** es un número que permite reconocer como único al elemento que es asignado. El identificador puede asignarse a un **proceso** o a un **evento**. Está representado por un círculo con el número de identificador en su interior para el caso de procesos, y con una letra mayúscula para el caso de eventos, como se muestra en la figura 9.

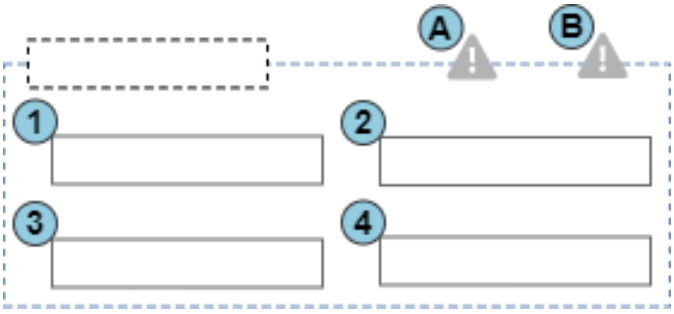

Figura 9-Identificadores en lenguaje SML

Regla:

Una **regla** es una sentencia de lógica booleana cuyas condiciones están determinadas en base al inicio de **procesos**, finalización de **procesos**, o *timeout* generado en la ejecución o en el llamado de un **proceso**. El resultado de la evaluación de la regla es la ejecución de llamadas a uno o varios **eventos**. Las reglas se representan gráficamente dentro de recuadros en la parte superior de un escenario, con la sentencia de condiciones en su interior izquierdo, y una breve descripción de la condición en un interior derecho, como se muestra en la figura 10.

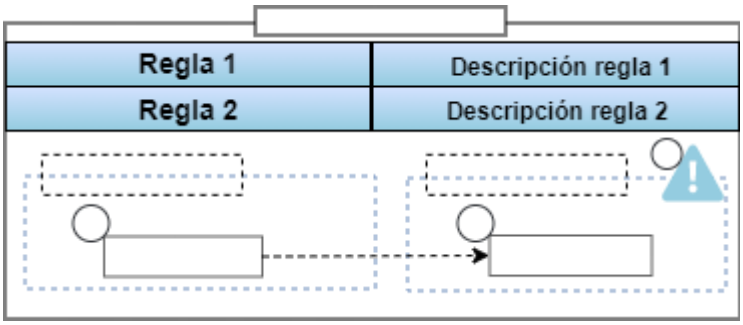

Figura 10-Regla en lenguaje SML

En un escenario pueden existir múltiples reglas. La sintaxis de una regla contiene una o varias condiciones y finaliza con el llamado a uno o varios eventos, como se muestra a continuación:

<condición 1>,<condición 2>, … , <condición N> : <evento 1> , <evento 2>, … , <evento N>

La sintaxis de una condición, dependiendo del caso se muestra en la tabla 1:

Tabla 1-Sintaxis de condiciones en lenguaje SML

Sintaxis	Descripción
<identificador> I	La condición es verdadera cuando el proceso del identificador ha iniciado
<identificador> F	La condición es verdadera cuando el proceso del identificador ha finalizado
<identificador> TO	La condición es verdadera cuando ha transcurrido el tiempo de *timeout* establecido para el proceso del identificador
<identificador A>TO<identificador B>	La condición es verdadera cuando ha transcurrido el tiempo de *timeout* establecido para el mensaje síncrono entre el proceso del identificador A y el proceso del identificador B.

A continuación, en la tabla 2 se presentan algunos ejemplos de reglas de monitoreo que se podrían definir con el lenguaje SML:

Tabla 2-Ejemplos de reglas de monitoreo con SML

Ejemplo	Regla
Se han ejecutado correctamente los procesos 2 y 3, sin embargo el proceso 7 ha presentado *timeout* en su ejecución. No se requiere invocar evento.	2I,2F,3I,3F,7TO
Si en algún punto de la ejecución, el proceso 5 y el proceso 8 presentan *timeout* en su ejecución, invocar al evento A.	5TO,8TO:A
Cuando los procesos 3 y 4 hayan finalizado, se requiere invocar al evento B.	3F,4F:B

Si se ejecutaron correctamente los procesos 4 y 5, y se presentó un *timeout* en la llamada desde el proceso 6 al proceso 8, invocar al evento C.	4I,4F,5I,5F,6TO8:C

Evento:

Un **evento** es una acción que será ejecutada en un **componente** tras cumplirse una determinada **regla**. El evento se representa gráficamente con un triángulo con un signo de exclamación en su interior y el identificador correspondiente, ubicado en la esquina superior derecha del componente que será llamado por la plataforma para ejecutar el evento definido, como se muestra en la figura 11.

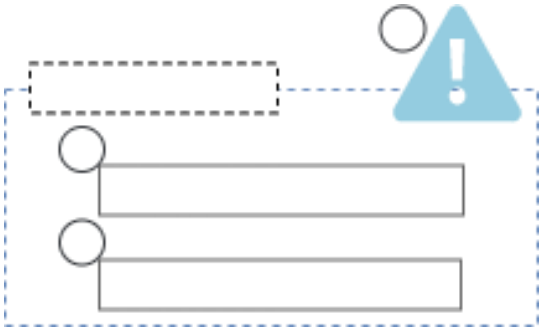

Figura 11-Evento en lenguaje SML

Mensajes:

Un **mensaje** representa la llamada desde un **proceso** a otro (por ejemplo un consumo de servicios web o comunicaciones TCP). Los mensajes pueden ser síncronos o asíncronos, siendo síncronos aquellos en los cuales el **proceso** que realiza la llamada se queda a la espera hasta obtener una respuesta del **proceso** destino o hasta cumplirse un tiempo de ***timeout*** definido, y los asíncronos aquellos donde el proceso de inicio hace la llamada al proceso destino y continúa con su ejecución, independientemente de haber recibido o no una respuesta.

Los **mensajes** síncronos son representados una línea contínua desde el **proceso** de inicio hacia el **proceso** destino, terminando con una punta de flecha. En un escenario de funcionamiento normal el proceso inicial envía un mensaje al proceso destino, el cual luego de un tiempo de procesamiento normal envía la respuesta al proceso inicial, continuando con el flujo.

Sin embargo, existen escenarios de monitoreo que requieren verificar fallas de comunicación, tiempos demasiado altos de respuesta o caídas en el servicio de algún componente o proceso. Para abordar estas situaciones se define un tiempo máximo de espera o *timeout* transcurrido desde que el proceso inicial envía un mensaje al proceso destino, hasta que recibe una respuesta. En la parte media de la flecha del mensaje se incluye un número en segundos que representa la cantidad de tiempo máxima establecida para considerar un mensaje sin respuesta como ***timeout*.**

Los **mensajes** asíncronos son representados por una línea entrecortada desde el **proceso** de inicio hacia el **proceso** destino, terminando con una punta de flecha. La representación gráfica de los mensajes síncronos y asíncronos se muestran en las figuras 12 y 13, respectivamente.

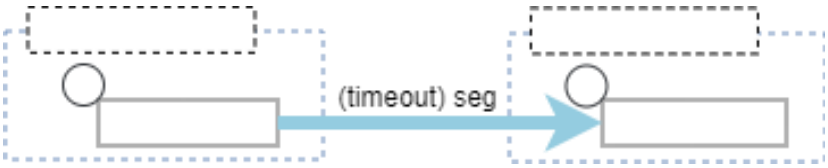

Figura 12-Mensaje síncrono en lenguaje SML

Figura 13-Mensaje asíncrono en lenguaje SML

Escenario:

Un **escenario** representa una situación o caso específico de los sistemas a monitorear. Es el contenedor de todos los elementos descritos en las secciones anteriores. La representación gráfica de un escenario es un recuadro con línea contínua, que contiene en la parte superior el nombre que describe a dicho escenario, como se muestra en la figura 14.

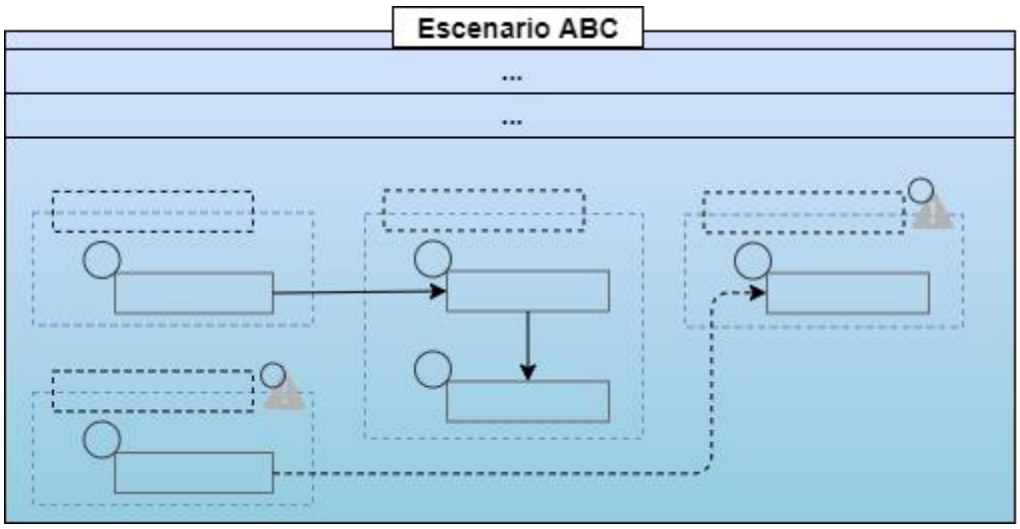

Figura 14-Escenario en lenguaje SML

Sintaxis concreta

En este paso se deben tener en cuenta todas las consideraciones importantes para la definición textual o gráfica del *DSML*. La propuesta de este estudio está centrada en el lenguaje SML para la elaboración de modelos de especificación de sistemas de monitoreo. Al ser un lenguaje de

modelado, tiene como característica fundamental guardar independencia entre el modelo y la implementación o detalles de bajo nivel. Por este motivo, el lenguaje ha sido elaborado con un alto nivel de abstracción, permitiendo que la implementación se la realice de forma independiente. A continuación, se mencionan ciertas consideraciones que se deberían tener para la generación de la plataforma de monitoreo en base al modelo SML.

La plataforma de monitoreo deberá ser accesible para todos los componentes del modelo. En una arquitectura de sistemas con componentes heterogéneos, la mejor alternativa para permitir la integración es utilizar tecnologías estándar. Es por esto que la plataforma de monitoreo deberá desplegar todos los servicios de monitoreo en diferentes presentaciones (*REST*, *SOAP*, comunicaciones *TCP*) permitiendo que componentes con diferentes tecnologías puedan integrarse al sistema de monitoreo. De la misma forma, se requiere que los componentes tengan la capacidad de comunicarse a través de la red hacia la plataforma de monitoreo, y en caso necesario exponer servicios para recibir notificaciones de eventos desde la plataforma de monitoreo.

Por otro lado, todos los procesos que se hayan definido en los componentes se comunicarán con la plataforma de monitoreo para notificar los cambios en su ciclo de vida (Inicio de proceso, Fin de proceso). Para el ciclo de vida de cada proceso y cada mensaje enviado entre procesos, la plataforma analizará el tiempo transcurrido desde el inicio del ciclo o del envío del mensaje, de manera que se pueda determinar cuando exista un *timeout*. En caso de cumplirse un *timeout* se establecerá como estado NO FINALIZADO al proceso en cuestión, lo cual permitirá la evaluación de las reglas para condiciones de no finalización de procesos.

En cada una de estas notificaciones la plataforma deberá evaluar las reglas definidas en los escenarios de monitoreo para ejecutar los llamados a cada evento asociado en las reglas. Por esta razón, es importante que en los componentes se deberán especificar únicamente los procesos que se deseen monitorear. En la práctica, es factible agrupar un conjunto de funciones o métodos de la ejecución interna de cada componente, y proyectarlos como una representación abstracta de un proceso o unidad de trabajo.

El esquema de mensajería debe utilizar rigurosamente marcas de tiempo, para lo cual los componentes deberán estar sincronizados en su totalidad con la plataforma de monitoreo. Las marcas de tiempo deberán ser incluidas en cada punto de las comunicaciones entre los componentes hacia la plataforma de monitoreo. Adicionalmente, se deberá controlar cada ejecución de los procesos mediante un identificador único universal (UUID), esto para permitir la trazabilidad en el flujo de procesamiento y comunicaciones en los sistemas.

En la instanciación de cada modelo, se deberá proveer toda la información relacionada al escenario en cuestión que permitirá la ejecución de la plataforma, incluyendo: direcciones IP, *hostname*, puertos de comunicación, protocolos soportados, parámetros de *timeout*, enlace o *listener* para las llamadas asociadas a los eventos en cumplimiento de una regla, entre otros.

Enfoque a servicios de monitoreo en la nube

En el día a día, existen muchos sistemas de diferentes empresas o instituciones que se comunican entre sí por medio de la red para diversos propósitos. En la práctica, el alcance del monitoreo realizado a dichos sistemas es individual y en algunos casos insuficiente. Esto genera carga operativa para el personal de producción, incertidumbre en los clientes finales cuando se presentan incidentes, posibles controversias entre las empresas e instituciones por temas legales o de privacidad de datos y dificultad para las entidades de regulación o auditoría, al encontrar fuentes de datos diferentes, implementadas con diferentes tecnologías y por diferentes proveedores. En este sentido, SML se proyecta hacia el modelado de sistemas de monitoreo en soluciones multi-empresariales, las cuales al ser desplegadas en la nube permiten a las empresas aprovechar todas las ventajas de *cloud computing* con respecto al monitoreo de sus sistemas en cuestión.

El nivel de granularidad que se desee utilizar, o el alcance de la especificación de los sistemas de monitoreo dependerá mucho de los intereses y necesidades de las empresas. Sin embargo, resulta interesante mencionar que con SML se podrían generar soluciones de monitoreo desde un punto de vista de un tercero, es decir, delegar las funciones de monitoreo a una empresa, proveedor o ente de regulación y su alcance puede llegar a ser incluso hasta sistemas de sistemas, dando lugar a nuevas posibilidades y soluciones de software.

Capítulo 4 – Validación

Este capítulo corresponde a la validación de la propuesta de este trabajo de titulación. El lenguaje SML es aplicado a un caso concreto de la industria del software en una empresa local para conocer los resultados con respecto a la capacidad del lenguaje de modelado para expresar las necesidades de monitoreo de una empresa.

Definición del caso de estudio

A continuación se detalla la conformación del caso de estudio, definiendo objetivos, recursos de información y medios para la ejecución del caso de estudio, utilizando las directrices para casos de estudio propuestas por Runeson & Höst (2009) en el ámbito de la ingeniería de software.

Objetivos

En el capítulo 3 se ha descrito a detalle el lenguaje de modelado SML, con todas las características gráficas correspondientes para modelar sistemas de monitoreo, dotando a la expresión de modelado con sintaxis específica y un sentido semántico. Es necesario validar que el lenguaje propuesto en este trabajo de titulación permite modelar las necesidades de monitoreo planteadas para una situación particular en la industria del software, manteniendo los objetivos alineados a los propósitos planteados en el Capítulo 1. En este sentido, por medio de un caso de estudio se pretende validar que SML permite expresar las necesidades de monitoreo de sistemas de forma gráfica y con una transformación de modelo a texto (M2T) desplegar servicios de monitoreo en la nube acorde a las necesidades establecidas en el modelo.

Ámbito de la investigación

Con este caso de estudio se pretende analizar el resultado de la utilización del lenguaje de modelado SML para expresar necesidades de monitoreo en un contexto definido. Específicamente, se busca determinar si el lenguaje de modelado tiene la suficiente capacidad de expresión para modelar las necesidades empresariales, analizando los resultados con respecto a tiempo requerido.

Conjunto de datos

De acuerdo a lo planteado en la metodología para este proyecto, se tiene como alcance únicamente una validación inicial del proyecto, por lo cual se considerara como fuente de información los datos proporcionados por la empresa, junto con la observación directa durante el proceso de diseño e implementación del modelo de monitoreo por el personal de la empresa. En esta etapa se recopilarán datos que permitan determinar los beneficios o debilidades encontradas en la aplicación del caso de estudio, con respecto a incorporación plataforma de monitoreo versus el proceso operativo manual de monitoreo, para los diferentes escenarios.

De la misma forma, la empresa proporcionará los datos estimados del tiempo o esfuerzo necesario para especificar, diseñar, desarrollar y desplegar un sistema de monitoreo para las necesidades de la empresa. Se contrastaran dichos valores con los resultados finales de tiempo obtenidos tras la implementación del caso de estudio en la empresa, para estimar la factibilidad de aplicación de esta propuesta como solución a las necesidades de monitoreo.

Selección

La empresa ha sido seleccionada en base a la arquitectura que soporta los servicios o transacciones de sus sistemas o aplicaciones de software, debiendo disponer de múltiples servicios desplegados en diferentes servidores que se comunican entre sí. Por este motivo, se ha seleccionado a la empresa Copicomp Cia. Ltda., quienes se han encontrado disponibles para la aplicación del caso de estudio y han presentado una necesidad de monitoreo candidata para el análisis.

Consideraciones éticas

La empresa Copicomp Cia. Ltda. pone a disposición de la investigación el equipo de desarrollo necesario para aplicar el lenguaje de modelado SML en un proceso definido de sus sistemas de software. Copicomp Cia. Ltda. se reserva el derecho de confidencialidad con respecto a los datos, información generada en la utilización de los sistemas y códigos fuente.

Métricas

El conjunto de datos que se ha definido para este caso de estudio permite analizar los datos de forma cuantitativa y obtener afirmaciones cualitativas con respecto al caso de estudio. Este caso de estudio tiene como propósito validar si el lenguaje de modelado propuesto es capaz de expresar las necesidades de monitoreo en un caso empresarial particular. La unidad de medida de tiempo a utilizarse para el análisis de validez será la hora.

Ejecución del caso de estudio

La empresa Copicomp Cia. Ltda. está ubicada en la ciudad de Cuenca y presta sus servicios desde el año 1999, y actualmente es proveedora de software para el sector público y privado. Para este caso de estudio se ha elegido el proceso de registro de los clientes que acceden al sitio web de la empresa para crear una cuenta de usuario en la plataforma Novelty de la empresa.

El proceso de registro inicia con el acceso del cliente al sitio web de la empresa en la sección Registrarse. Como primer paso el cliente ingresa la información de contacto correspondiente y da clic en Registrarse, como se muestra en la figura 15. La aplicación de registro valida la información ingresada y almacena al nuevo cliente en la base de datos.

En el segundo paso el cliente ingresa la información de su empresa, la cual será asociada a la cuenta de usuario para la utilización de la plataforma Novelty, como se observa en la figura 16. Al dar clic en Crear Cuenta la aplicación de registro se comunicará con los servicios web para creación de cuentas de usuario, los cuales validarán toda la información ingresada y procederán a crear la cuenta de usuario, respondiendo a la aplicación de registro la confirmación correspondiente.

En el tercer paso se redirige al usuario a la aplicación de suscripciones, la cual permite seleccionar el tipo de paquete de transacciones a utilizar en la plataforma, y realizar el pago respectivo en línea. Para efectuar dicha transacción se utiliza la pasarela de pagos Payphone (marca registrada), la cual es una plataforma de pagos y cobros creada en Ecuador, que permite realizar transacciones con tarjetas de crédito y débito mediante un teléfono celular. La sección de suscripciones se muestra en la figura 17.

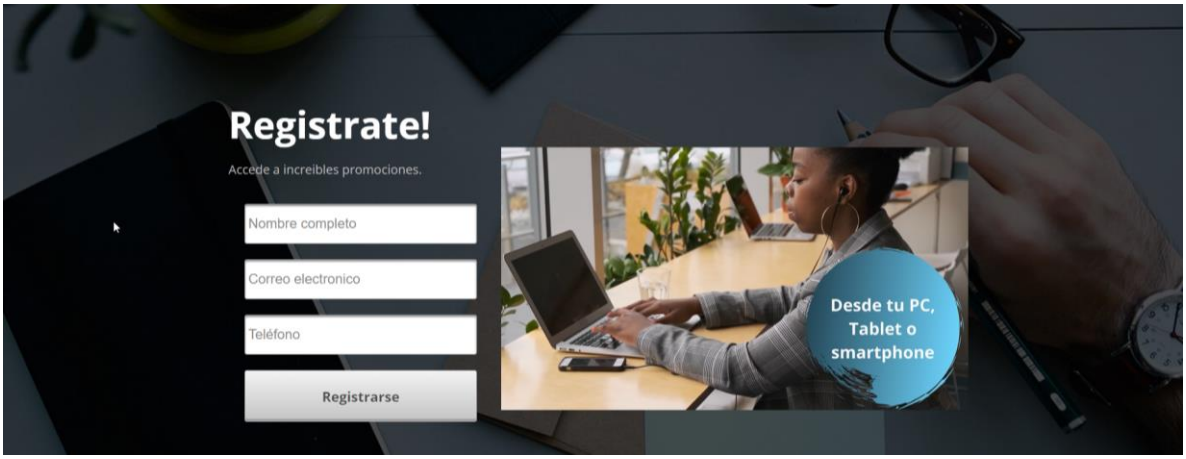

Figura 15-Formulario de registro de datos de contacto

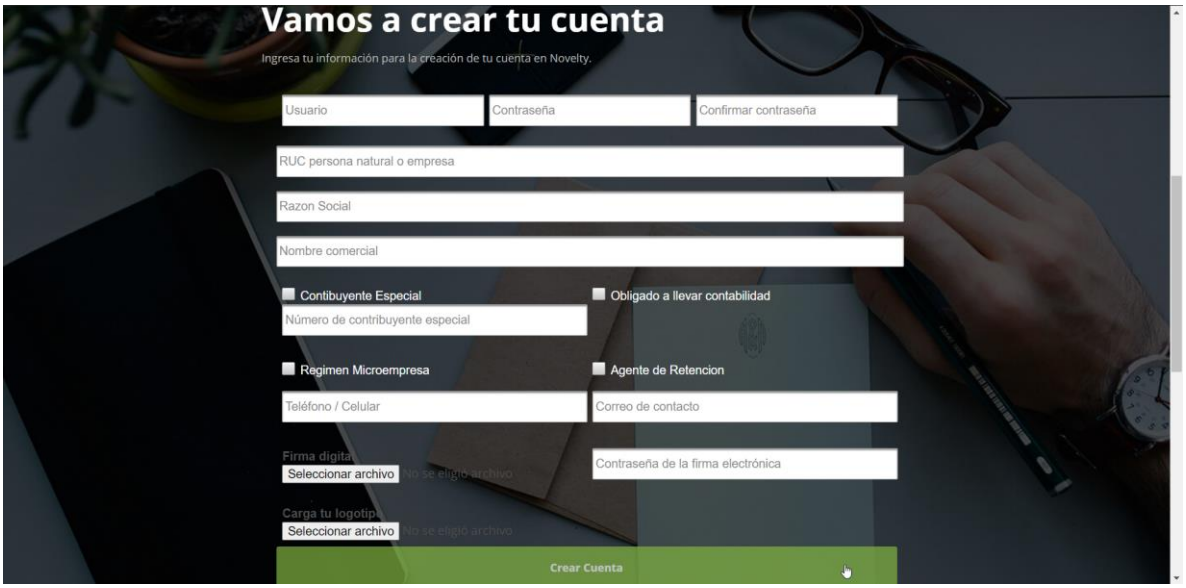

Figura 16-Formulario de datos de empresa

Figura 17-Interfaz de suscripción del cliente

Para la empresa resulta muy importante mantener siempre disponible para los clientes el canal de registro y suscripción para la plataforma, y conocer de primera mano cualquier eventualidad. Por este motivo, la empresa requiere monitorear todo el proceso de registro del cliente en sus múltiples variantes o escenarios, como se define en la tabla 3.

Tabla 3-Escenarios de monitoreo requeridos por la empresa

Escenario	Descripción
Registro y suscripción completa	El cliente accede al sitio web, inicia el proceso de registro, ingresa la información de contacto, ingresa la información de cuenta, selecciona el paquete de comprobantes y finaliza su proceso de registro y suscripción.
Registro sin suscripción	El cliente accede al sitio web, inicia el proceso de registro, ingresa la información de contacto, ingresa la información de cuenta y al concluir su creación de cuenta finaliza el proceso sin suscribirse.
Error inesperado en pasarela de pago	El cliente accede al sitio web, inicia el proceso de registro, ingresa la información de contacto, ingresa la información de cuenta y se registra, selecciona el paquete de comprobantes y durante el procesamiento del pago con la pasarela Payphone se produce un error, dejando incompleto su proceso de suscripción.
Error en servicios web para creación de cuenta	El cliente accede al sitio web, inicia el proceso de registro, ingresa la información de contacto, ingresa la información de cuenta y durante el procesamiento de creación de cuenta se produce un error, dejando incompleto su proceso de registro.

Para hacer posible este caso de estudio, se ha considerado conveniente implementar un prototipo de herramienta de modelado basado en el lenguaje SML. La herramienta ha sido desarrollada utilizando el entorno de trabajo *Eclipse OBEO* de ObeoSoft (2021), el cual incluye gran variedad de capacidades y componentes para diseño y generación de código basado en modelos. Disponer de este prototipo permite materializar las ideas que son expresadas en los diferentes modelos, los cuales estarán de acuerdo a las reglas sintácticas y semánticas que han sido definidas en el lenguaje.

Para la construcción del prototipo se ha definido un modelo UML con el módulo *Eclipse Modeling Framework (EMF)* de ObeoSoft (2021) en base a la etapa de captura del conocimiento del dominio, como se muestra en la figura 18. En la parte central se observa el diagrama de dominio definido para generar el prototipo y en la parte izquierda se observan las clases generadas por el EMF. Con el dominio definido, se ha utilizado el módulo *Eclipse Sirius de* ObeoSoft (2021) para generar un editor gráfico y diseñar la herramienta de modelado. Se han definido los elementos gráficos que conforman el lenguaje SML para utilizarlos en la paleta de diseño, de acuerdo a lo establecido para el lenguaje en la sección de sintaxis concreta del capítulo anterior. En la figura 19 se observa la jerarquía de los elementos en la parte central, en la parte izquierda se observan los archivos de código fuente generado que soporta al diseño, en la parte inferior se visualizan todas las propiedades correspondientes a los elementos, y en la parte derecha se muestra una pre-visualización del modelo generado en base a lo definido en el entorno de trabajo.

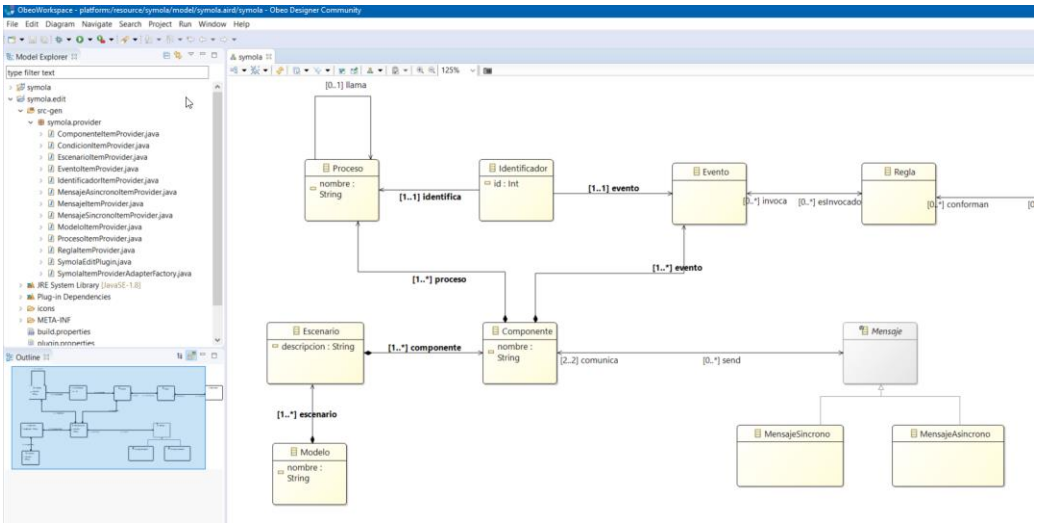

Figura 18-Modelo de dominio con Eclipse Modeling Framework

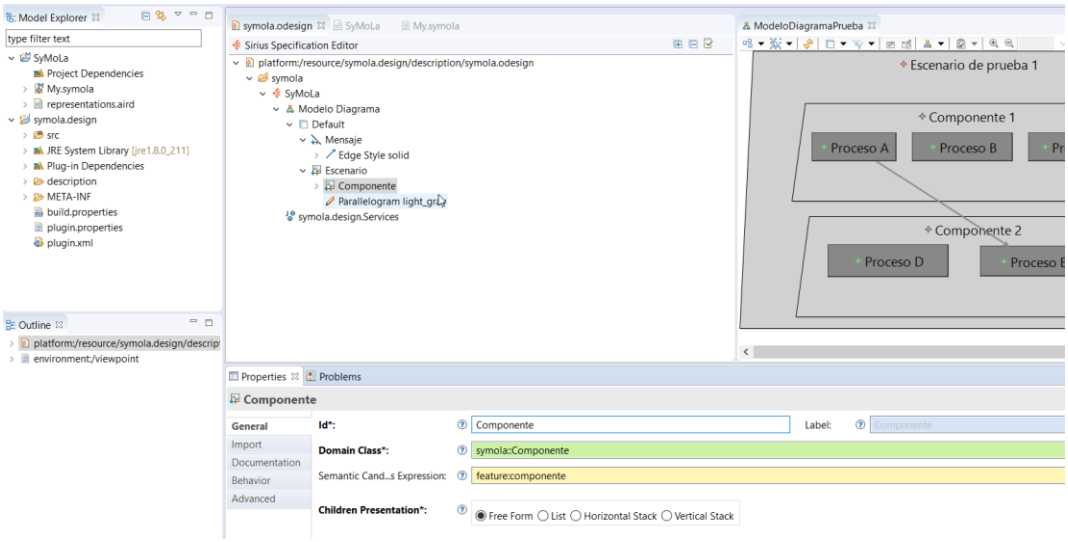

Figura 19-Diseño de elementos gráficos del lenguaje SML con OBEO Designer

De forma complementaria, para alcanzar el objetivo del caso de estudio y validar los beneficios o posibles debilidades del lenguaje se ha incluido en el prototipo una herramienta de transformación modelo a texto (M2T), la cual permite convertir el modelo gráfico de monitoreo en código fuente para ser utilizado como base para desplegar servicios en la plataforma de monitoreo en la nube. Los diagramas generados con la herramienta de modelado son representados en el módulo *Eclipse Sirius* de ObeoSoft (2021) por medio de un archivo XML. El prototipo hace la lectura del modelo y realiza la transformación hacia código fuente en el lenguaje de programación Java, el cual cuenta con artefactos para ser incorporados en un proyecto web y su posterior despliegue en un entorno *cloud computing*. La figura 20 muestra el archivo XML que describe el modelo generado en el editor gráfico. La figura 21 muestra el código fuente generado tras la transformación, la cual incluye servicios web RESTful con interfaces específicas para la comunicación de cada componente

del modelo, clases asociadas al dominio del monitoreo de sistemas, algoritmos de validación de condiciones y reglas.

symola.odesign SyMoLa My.symola

```
<?xml version="1.0" encoding="UTF-8"?>
<symola:Modelo xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:symola="http://www.example.org/s
  <escenario descripcion="Escenario de prueba 1">
    <componente nombre="Componente 1">
      <proceso nombre="Proceso A" llama="//@escenario.0/@componente.1/@proceso.1"/>
      <proceso nombre="Proceso B"/>
      <proceso nombre="Proceso C"/>
    </componente>
    <componente nombre="Componente 2">
      <proceso nombre="Proceso D"/>
      <proceso nombre="Proceso E"/>
    </componente>
  </escenario>
  <escenario descripcion="Escenario alternativo">
    <componente nombre="Componente 3">
      <proceso nombre="Proceso F"/>
    </componente>
    <componente nombre="Componente 4">
      <proceso nombre="Proceso G" llama="//@escenario.1/@componente.1/@proceso.1"/>
      <proceso nombre="Proceso H"/>
    </componente>
  </escenario>
</symola:Modelo>
```

Figura 20-Representación XML del modelo gráfico con SML

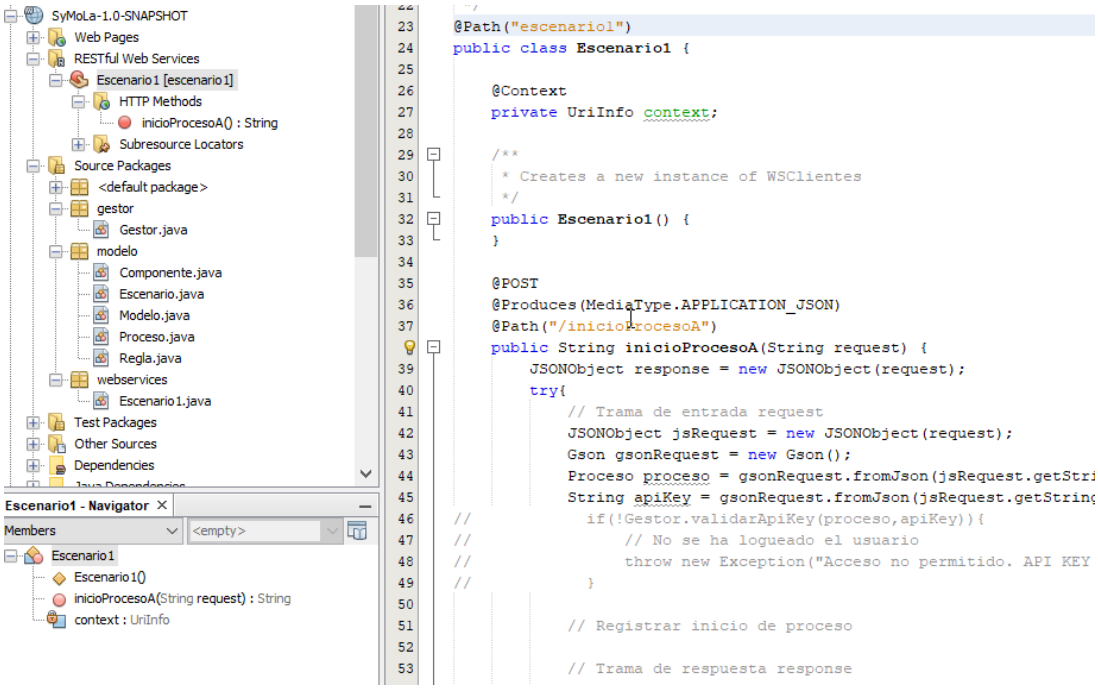

Figura 21-Código fuente generado con herramienta M2T

Bajo este contexto, se inicia el caso de estudio capacitando al personal designado para la ejecución del modelado de las necesidades de monitoreo utilizando el lenguaje SML. Con el editor gráfico de la herramienta, el personal designado procede a definir el modelo, el cual contiene los 4 escenarios de monitoreo definidos en la tabla 3 para cada variante del proceso de registro de

clientes. Como resultado se tienen definidos 4 escenarios: registro y suscripción completa, registro sin suscripción, error inesperado en pasarela de pago, y error en servicios web para creación de cuenta.

En el escenario de registro, validación de datos y suscripción completa, se plantean como componentes: **Aplicación de registro, Servicios web Novelty, Servicios web pagos**. En la Aplicación de registro, se inicia con el proceso **Registro datos de contacto**, una vez culminado este proceso se inicia con el proceso de **Registro cuenta de empresa**, el cual hace el llamado al proceso **Creación de cuenta**, el cual a su vez llama a los procesos internos de **Validación de datos, creación de empresa**, finalizando con el proceso inicial de **Registro cuenta de empresa**. Luego, la Aplicación de registro inicia el proceso de **Pago mediante pasarela**, el cual hace el llamado al proceso **Pago Payphone** del componente Servicios web pagos, el cual finaliza el flujo completo del escenario. De la misma forma, en este escenario se incluye el caso alternativo, en el cual el proceso **Validación de datos** no culmina totalmente, desencadenando al proceso interno **Datos inválidos** y finalizando el flujo de escenario. Las reglas establecidas para este caso se muestran a continuación en la tabla 4.

Tabla 4-Reglas escenario de registro, validación y suscripción completa

Regla	Descripción
1F , 2I , 3I , 4I , 5I	Flujo con datos de empresa inválidos
1F , 3F , 4F , 6F , 2F , 8F , 7F : A	Flujo con datos de empresa validados correctamente

En el escenario de registro sin suscripción, se plantean como componentes: **Aplicación de registro**, **Servicios web novelty**. En la Aplicación de registro, se inicia con el proceso **Registro datos de contacto**, una vez culminado este proceso se inicia con el proceso de **Registro cuenta de empresa**, el cual hace el llamado al proceso **Creación de cuenta**, el cual a su vez llama a los procesos internos de **Validación de datos**, **Creación de empresa**, finalizando con el proceso inicial de **Registro cuenta de empresa**. Luego, la Aplicación de registro inicia el proceso de **Pago mediante pasarela**, sin embargo el cliente desiste de realizar el proceso de suscripción, con lo cual se alcanza el tiempo definido de *timeout* para dicho proceso y se finaliza el flujo del escenario. Las reglas establecidas para este caso se muestran a continuación en la tabla 5.

Tabla 5-Reglas escenario de registro sin suscripción

Regla	Descripción
1F , 3F , 4F , 6F , 2F , 7I , 7TO : B	Flujo de registro con timeout en pago mediante pasarela

En el escenario de error inesperado en pasarela de pago, se plantean como componentes: Aplicación de registro, Servicios web novelty, Servicios web pagos. En la Aplicación de registro, se inicia con el proceso **Registro datos de contacto**, una vez culminado este proceso se inicia con el proceso de **Registro cuenta de empresa**, el cual hace el llamado al proceso **Creación de cuenta**, el cual a su vez llama a los procesos internos de **Validación de datos**, **Creación de empresa**, finalizando con el proceso inicial de **Registro cuenta de empresa**. Luego, la Aplicación de registro inicia el proceso de **Pago mediante pasarela**, se hace el llamado al proceso de **Pago Payphone** del componente Servicio Web Pagos el cual inicia normalmente, sin embargo presenta un error inesperado al conectar con la pasarela de pagos Payphone, por ende no se culmina el proceso

normal, alcanzando un tiempo de timeout definido para el proceso **Pagos Payphone** y finalizando el flujo de escenario sin haber realizado el pago de suscripción. De la misma forma en este escenario se monitorea el caso alternativo, en el cual el proceso **Pago mediante pasarela** del componente Aplicación de Registro hace el llamado al proceso **Pago Payphone** del componente Servicios Web Pagos, y no recibe respuesta en un tiempo determinado, alcanzando un timeout y finalizando el flujo de escenario sin realizar la suscripción. Las reglas establecidas para este caso se muestran a continuación en la tabla 6.

Tabla 6-Reglas escenario de error inesperado en pasarela de pago

Regla	Descripción
1F , 3F , 4F , 6F , 2F , 7I , 8TO : C	Flujo error de Servicio Web Pagos conectando con Payphone
1F , 3F , 4F , 6F , 2F , 7I , 7TO8 : D	Flujo timeout en respuesta de Servicio Web Pagos

En el escenario de error en servicios web para creación de cuenta, se plantean como componentes: **Aplicación de registro**, **Servicios web novelty**. En la Aplicación de registro, se inicia con el proceso **Registro datos de contacto**, una vez culminado este proceso se inicia con el proceso de **Registro cuenta de empresa**, el cual hace el llamado al proceso **Creación de cuenta**, el cual a su vez llama a los procesos internos de **Validación de datos**, **Creación de cuenta**, sin embargo se presenta algún error interno en el transcurso de la ejecución del proceso **Creación de empresa** generando un timeout, con lo cual se finaliza el flujo del escenario sin culminar con la creación de cuenta de la empresa. De la misma forma, se plantea el caso alternativo en el cual el proceso **Registro cuenta de empresa** del componente Aplicación de registro hace el llamado al proceso **Creación de cuenta** del componente Servicios web Novelty, sin embargo no recibe respuesta durante el tiempo establecido, generando un timeout con lo cual finaliza el flujo del escenario sin culminar con la creación de la cuenta.

Tabla 7-Reglas escenario de error en servicios web para creación de cuenta

Regla	Descripción
1F , 2I , 3I , 3TO : E	Flujo de registro con timeout por error interno en creación de cuenta
1F , 2I , 2TO3 : F	Flujo de registro con timeout en respuesta de servicios para creación de cuenta

A continuación con las figuras 22, 23, 24 y 25 se muestran los cuatro escenarios de monitoreo planteados, expresados mediante el lenguaje de modelado SML.

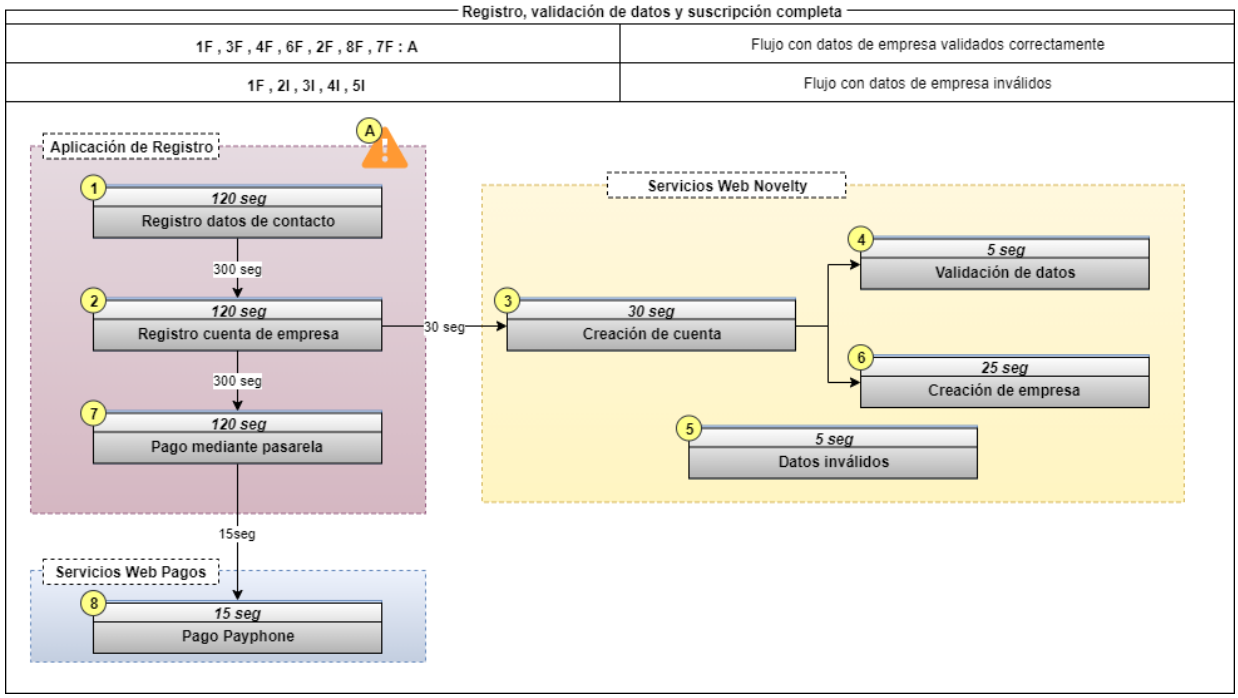

Figura 22-Modelo SML escenario de registro, validación de datos y suscripción completa

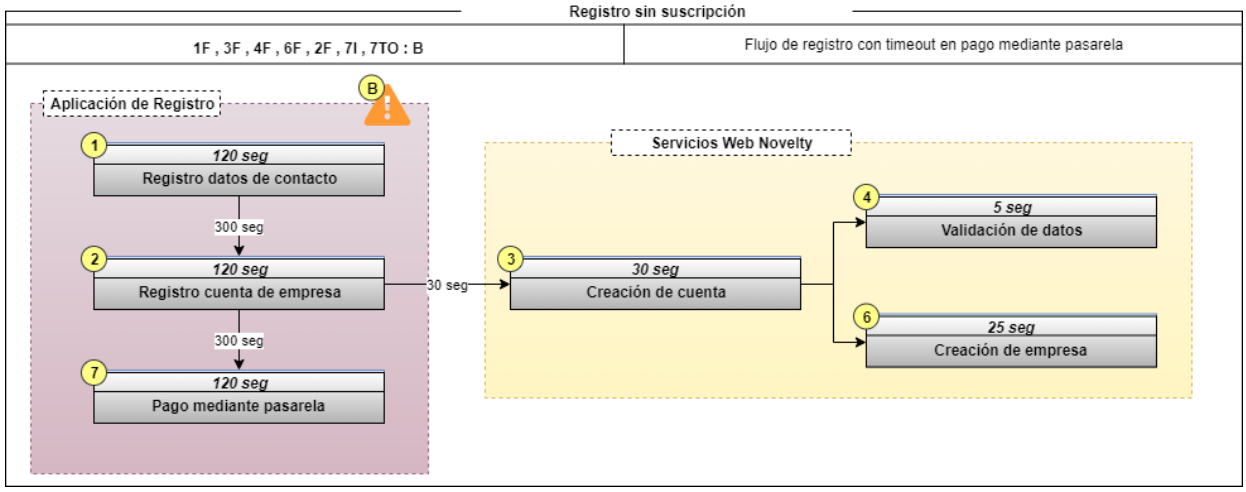

Figura 23-Modelo SML escenario de registro sin suscripción

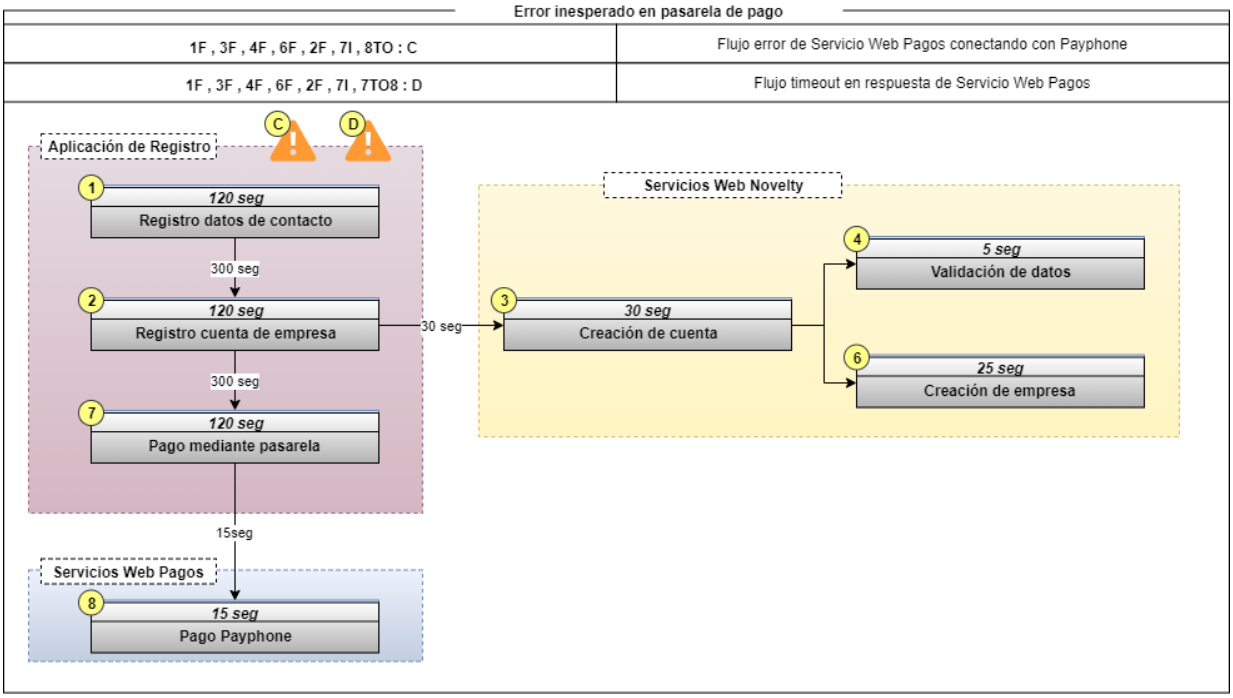

Figura 24-Modelo SML escenario de error inesperado en pasarela de pago

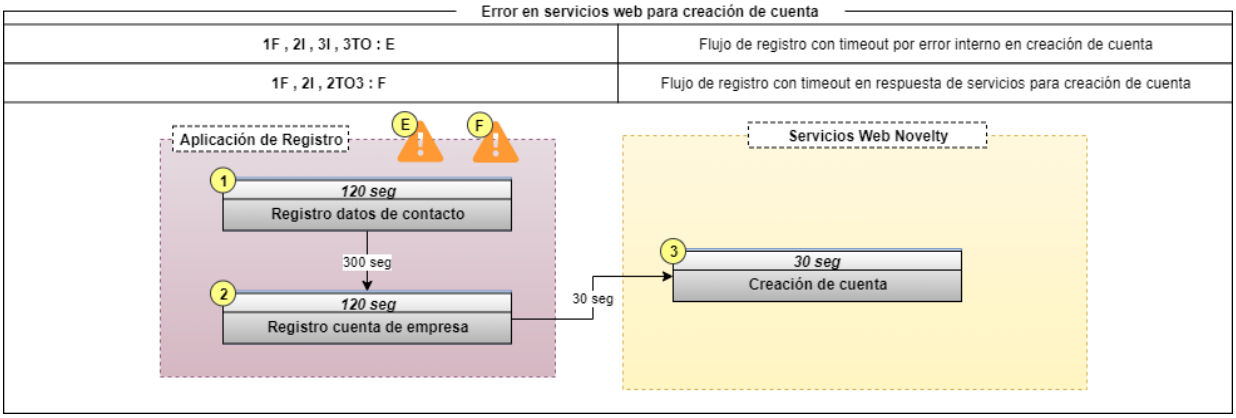

Figura 25-Modelo SML escenario de error en servicios web creación de cuenta

Una vez se dispone del modelo de monitoreo, se procede a transformarlo a código fuente utilizando la herramienta de transformación. Como resultado se han generado 4 servicios web de tipo RESTful, correspondientes a cada escenario de monitoreo. Se han generado las clases de dominio de monitoreo que permiten instanciar los elementos especificados en el modelo. De la misma forma, se han generado clases auxiliares que permiten ejecutar hilos de control para verificar ciclos de vida de cada proceso y actualizar las condiciones que serán evaluadas en las reglas definidas. Finalmente se generan secciones de código de consumo de los servicios web desde el lado del cliente para cada uno de los procesos.

Con el código fuente generado se ha conformado una aplicación web denominada *SyMoLa-1.0-SNAPSHOT* que corresponde a la plataforma de monitoreo a ser desplegada en la nube. La plataforma consta de 4 servicios web correspondientes a cada escenario de monitoreo planteado de acuerdo a los requerimientos. La plataforma ha sido desplegada en un servidor de aplicaciones Wildfly sobre la infraestructura de la empresa y expone sus servicios de forma local para los componentes en estudio. Por otro lado, el equipo de desarrollo ha incorporado las secciones de código fuente cliente en los componentes Aplicación de registro, Servicios web Novelty, Servicios web pagos para las comunicaciones hacia la plataforma de monitoreo.

Para disponer de un entorno de monitoreo en ejecución en base a los requerimientos de la empresa, se procede a preparar cada uno de los escenarios de monitoreo de acuerdo a lo establecido, permitiendo que se reproduzca el flujo de procesos, comunicaciones, validación de reglas y eventos. Se ha propuesto la realización de pruebas controladas en cada uno de estos escenarios, designando como cliente al personal de la empresa y utilizando datos de prueba en los procesos de registro y creación de cuenta, manteniendo una bitácora de las pruebas realizadas para la posterior verificación en los servidores y bases de datos de la empresa.

En el escenario de **Registro, validación de datos y suscripción completa** se han levantado todos los servicios involucrados en el proceso de registro de los clientes. Se ha solicitado al cliente ingresar los datos incorrectos para la primera regla, e ingresar los datos correctos para la segunda regla. En las tablas 8 y 9 se ha resumido la línea de tiempo del flujo completo del escenario planteado desde la perspectiva funcional del sistema versus la plataforma de monitoreo, para los dos casos de reglas planteados.

Tabla 8-Línea de tiempo datos de empresa inválidos

Marca de tiempo	Perspectiva	Detalle
08/03/2021 09:24:11	Funcional	El cliente accede a la sección de registro.
08/03/2021 09:24:12	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 09:24:52	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 09:24:52	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 09:24:55	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 09:24:55	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 09:28:31	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta

08/03/2021 09:28:32	Plataforma	Inicio del proceso (3I) – Creación de cuenta. Control de timeout (2TO3), esperando respuesta de proceso 3. Control de timeout (3TO).
08/03/2021 09:28:33	Plataforma	Inicio del proceso (4I) – Validación de datos. Control de timeout (4TO).
08/03/2021 09:28:33	Plataforma	Inicio del proceso (5I) – Datos inválidos Control de timeout (5TO). Fin del proceso (5I) – Datos inválidos. Fin control de timeout (5TO) OK.
08/03/2021 09:28:34	Plataforma	Fin del proceso (4F) – Validación de datos. Fin control de timeout (4TO) OK.
08/03/2021 09:28:34	Plataforma	Fin de proceso (3F) – Creación de cuenta Fin control de timeout (3TO).
08/03/2021 09:28:35	Plataforma	Fin control de timeout (2TO3). Fin de proceso (2F) – Registro cuenta de empresa Fin control de timeout (2TO).
08/03/2021 09:28:36	Funcional	El cliente recibe un mensaje de error indicando que los datos ingresados no son válidos.

Tabla 9-Línea de tiempo datos de empresa validados correctamente

Marca de tiempo	Perspectiva	Detalle
08/03/2021 09:34:41	Funcional	El cliente accede a la sección de registro.
08/03/2021 09:34:41	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 09:35:03	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 09:35:03	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 09:35:05	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 09:35:05	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 09:37:10	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta
08/03/2021 09:37:11	Plataforma	Control de timeout (2TO3), esperando respuesta de proceso 3. Inicio del proceso (3I) – Creación de cuenta. Control de timeout (3TO).
08/03/2021 09:37:12	Plataforma	Inicio del proceso (4I) – Validación de datos. Control de timeout (4TO). Fin del proceso (4F) – Validación de datos. Fin control de timeout (4TO) OK.

08/03/2021 09:37:13	Plataforma	Inicio del proceso (6I) – Creación de empresa. Control de timeout (6TO). Fin del proceso (6F) – Creación de empresa. Fin control de timeout (6TO) OK.
08/03/2021 09:37:13	Plataforma	Fin del proceso (3F) – Creación de cuenta. Fin control de timeout (3TO).
08/03/2021 09:37:21	Plataforma	Fin de proceso (2F) – Registro cuenta de empresa. Fin control de timeout (2TO). Control de timeout (2TO7).
08/03/2021 09:37:23	Funcional	El cliente recibe un mensaje de confirmación de que su cuenta ha sido creada correctamente. La aplicación de registro muestra la sección de suscripción con las diferentes alternativas de compra.
08/03/2021 09:37:24	Plataforma	Fin control de timeout (2TO7) OK. Inicio del proceso (7I) – Pago mediante pasarela. Control de timeout (7TO)
08/03/2021 09:39:33	Funcional	El cliente selecciona la opción preferida de suscripción, ingresa la información solicitada para la realización del pago con Payphone y confirma la información a enviar.
08/03/2021 09:39:33	Plataforma	Control de timeout (7TO8).
08/03/2021 09:39:34	Plataforma	Inicio del proceso (8I) – Pago Payphone. Control de timeout (8TO). Fin del proceso (8F) – Pago Payphone. Fin control de timeout (8TO).
08/03/2021 09:41:15	Funcional	Los servicios de pago procesan la información ingresada y realizan la transacción hacia la pasarela de pagos Payphone. Se muestra al cliente un mensaje de confirmación sobre el pago realizado.
08/03/2021 09:41:15	Plataforma	Fin control de timeout (7TO8). Fin de proceso (7I) – Pago mediante pasarela.

Para el escenario de **Registro sin suscripción** se ha solicitado al cliente realizar el flujo normal de registro y en el proceso de **Pago mediante pasarela** cerrar el formulario sin realizar la suscripción. En la tabla 10 se ha resumido la línea de tiempo del flujo completo del escenario desde la perspectiva funcional del sistema versus la plataforma de monitoreo.

Tabla 10-Línea de tiempo registro con timeout en pago mediante pasarela

Marca de tiempo	Perspectiva	Detalle
08/03/2021 10:03:35	Funcional	El cliente accede a la sección de registro.
08/03/2021 10:03:36	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 10:04:13	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.

08/03/2021 10:04:20	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 10:04:21	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 10:04:21	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 10:05:13	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta
08/03/2021 10:05:13	Plataforma	Control de timeout (2TO3), esperando respuesta de proceso 3. Inicio del proceso (3I) – Creación de cuenta. Control de timeout (3TO).
08/03/2021 10:05:14	Plataforma	Inicio del proceso (4I) – Validación de datos. Control de timeout (4TO). Fin del proceso (4F) – Validación de datos. Fin control de timeout (4TO) OK.
08/03/2021 10:05:16	Plataforma	Inicio del proceso (6I) – Creación de empresa. Control de timeout (6TO). Fin del proceso (6F) – Creación de empresa. Fin control de timeout (6TO) OK.
08/03/2021 10:05:28	Plataforma	Fin del proceso (3F) – Creación de cuenta. Fin control de timeout (3TO).
08/03/2021 10:05:32	Plataforma	Fin de proceso (2F) – Registro cuenta de empresa. Fin control de timeout (2TO). Control de timeout (2TO7).
08/03/2021 10:05:34	Funcional	El cliente recibe un mensaje de confirmación de que su cuenta ha sido creada correctamente. La aplicación de registro muestra la sección de suscripción con las diferentes alternativas de compra.
08/03/2021 10:05:36	Plataforma	Fin control de timeout (2TO7) OK. Inicio del proceso (7I) – Pago mediante pasarela. Control de timeout (7TO)
08/03/2021 10:05:42	Funcional	El cliente desiste de realizar el ingreso de información para el pago y cierra el formulario de suscripción.
08/03/2021 10:07:42	Plataforma	Alcanza el tiempo de timeout especificado para el proceso (7TO). Desencadena el evento B del componente Aplicación de registro.

Para el escenario **Error inesperado en pasarela de pago** se ha deshabilitado temporalmente el acceso a los servicios de Payphone para el caso de la regla 1, y se ha inhabilitado los servicios web de pagos para la regla 2. En las tablas 11 y 12 se ha resumido la línea de tiempo del flujo completo del escenario desde la perspectiva funcional del sistema versus la plataforma de monitoreo.

Tabla 11-Línea de tiempo error de servicio web pagos conectando con Payphone

Marca de tiempo	Perspectiva	Detalle
08/03/2021 11:40:20	Funcional	El cliente accede a la sección de registro.
08/03/2021 11:40:20	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 11:40:45	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 11:40:46	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 11:40:48	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 11:40:49	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 11:43:49	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta
08/03/2021 11:43:49	Plataforma	Control de timeout (2TO3), esperando respuesta de proceso 3. Inicio del proceso (3I) – Creación de cuenta. Control de timeout (3TO).
08/03/2021 11:43:51	Plataforma	Inicio del proceso (4I) – Validación de datos. Control de timeout (4TO). Fin del proceso (4F) – Validación de datos. Fin control de timeout (4TO) OK.
08/03/2021 11:43:53	Plataforma	Inicio del proceso (6I) – Creación de empresa. Control de timeout (6TO). Fin del proceso (6F) – Creación de empresa. Fin control de timeout (6TO) OK.
08/03/2021 11:44:21	Plataforma	Fin del proceso (3F) – Creación de cuenta. Fin control de timeout (3TO).
08/03/2021 11:44:22	Plataforma	Fin de proceso (2F) – Registro cuenta de empresa. Fin control de timeout (2TO). Control de timeout (2TO7).
08/03/2021 11:44:24	Funcional	El cliente recibe un mensaje de confirmación de que su cuenta ha sido creada correctamente. La aplicación de registro muestra la sección de suscripción con las diferentes alternativas de compra.
08/03/2021 11:44:25	Plataforma	Fin control de timeout (2TO7) OK. Inicio del proceso (7I) – Pago mediante pasarela. Control de timeout (7TO)
08/03/2021 11:47:51	Funcional	El cliente selecciona la opción preferida de suscripción, ingresa la información solicitada para la realización del pago con Payphone y confirma la información a enviar.

08/03/2021 11:47:51	Plataforma	Control de timeout (7TO8).
08/03/2021 11:47:51	Plataforma	Inicio del proceso (8I) – Pago Payphone. Control de timeout (8TO).
08/03/2021 11:48:05	Plataforma	Alcanza el tiempo especificado de timeout (8TO). Fin control de timeout (7TO8). Fin de proceso (7F) – Pago mediante pasarela. Fin control de timeout (7TO)
08/03/2021 11:48:06	Funcional	Se muestra un mensaje de error al cliente, indicándole que los servicios no están disponibles por el momento.

Tabla 12-Línea de tiempo timeout en respuesta de servicio web pagos

Marca de tiempo	Perspectiva	Detalle
08/03/2021 12:24:37	Funcional	El cliente accede a la sección de registro.
08/03/2021 12:24:37	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 12:24:42	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 12:24:42	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 12:24:44	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 12:24:44	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 12:26:20	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta
08/03/2021 12:26:21	Plataforma	Control de timeout (2TO3), esperando respuesta de proceso 3. Inicio del proceso (3I) – Creación de cuenta. Control de timeout (3TO).
08/03/2021 12:26:23	Plataforma	Inicio del proceso (4I) – Validación de datos. Control de timeout (4TO). Fin del proceso (4F) – Validación de datos. Fin control de timeout (4TO) OK.
08/03/2021 12:26:23	Plataforma	Inicio del proceso (6I) – Creación de empresa. Control de timeout (6TO). Fin del proceso (6F) – Creación de empresa. Fin control de timeout (6TO) OK.
08/03/2021 12:26:31	Plataforma	Fin del proceso (3F) – Creación de cuenta. Fin control de timeout (3TO).
08/03/2021 12:26:31	Plataforma	Fin de proceso (2F) – Registro cuenta de empresa. Fin control de timeout (2TO). Control de timeout (2TO7).

08/03/2021 12:26:33	Funcional	El cliente recibe un mensaje de confirmación de que su cuenta ha sido creada correctamente. La aplicación de registro muestra la sección de suscripción con las diferentes alternativas de compra.
08/03/2021 12:26:34	Plataforma	Fin control de timeout (2TO7) OK. Inicio del proceso (7I) – Pago mediante pasarela. Control de timeout (7TO)
08/03/2021 12:28:03	Funcional	El cliente selecciona la opción preferida de suscripción, ingresa la información solicitada para la realización del pago con Payphone y confirma la información a enviar.
08/03/2021 12:28:03	Plataforma	Control de timeout (7TO8).
08/03/2021 12:28:19	Plataforma	Alcanza el tiempo especificado de timeout (7TO8). Fin de proceso (7F) - Pago mediante pasarela.
08/03/2021 12:28:20	Funcional	Se muestra un mensaje de error al cliente, indicándole que los servicios no están disponibles por el momento.

Para el escenario **Error en servicios web para creación de cuenta** se ha deshabilitado temporalmente un recurso mandatorio para la ejecución correcta del proceso de creación de cuenta, provocando un error interno y dando paso a la regla 1, y se ha inhabilitado los servicios web de novelty para la regla 2. En la tabla 13 se ha resumido la línea de tiempo del flujo completo del escenario desde la perspectiva funcional del sistema versus la plataforma de monitoreo.

Tabla 13-Línea de tiempo timeout por error interno en creación de cuenta

Marca de tiempo	Perspectiva	Detalle
08/03/2021 15:13:45	Funcional	El cliente accede a la sección de registro.
08/03/2021 15:13:46	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 15:14:02	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 15:14:02	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 15:14:08	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 15:14:08	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 15:15:31	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta
08/03/2021 15:15:32	Plataforma	Control de timeout (2TO3), esperando respuesta de proceso 3. Inicio del proceso (3I) – Creación de cuenta. Control de timeout (3TO).

08/03/2021 15:16:02	Plataforma	Alcanzado tiempo definido para timeout (3TO). Fin del proceso (3F) – Creación de cuenta
08/03/2021 15:16:02	Plataforma	Fin control de timeout (2TO3) Fin de proceso (2F) – Registro cuenta de empresa
08/03/2021 15:16:04	Funcional	Se presenta un mensaje de error al cliente, indicando que no ha sido posible crear la cuenta.

Tabla 14-Línea de tiempo timeout en respuesta de servicios para creación de cuenta

Marca de tiempo	Perspectiva	Detalle
08/03/2021 15:43:20	Funcional	El cliente accede a la sección de registro.
08/03/2021 15:43:21	Plataforma	Inicio del proceso (1I) – Registro datos de contacto. Control de timeout (1TO).
08/03/2021 15:43:33	Funcional	El cliente procede a llenar la información de contacto y confirma la información ingresada.
08/03/2021 15:43:33	Plataforma	Fin del proceso (1F) – Registro datos de contacto. Fin control de timeout (1TO) OK. Control de timeout (1TO2), esperando a Inicio proceso 2.
08/03/2021 15:43:35	Funcional	Se visualiza el formulario para ingreso de la información de cuenta de la empresa.
08/03/2021 15:43:36	Plataforma	Fin control de timeout (1TO2) OK. Inicio del proceso (2I) – Registro cuenta de empresa. Control de timeout (2TO).
08/03/2021 15:43:58	Funcional	El cliente ingresa la información relacionada a la empresa y envía la información. Espera un momento hasta que se culmine el proceso de creación de cuenta Control de timeout (2TO3), esperando respuesta de proceso 3.
08/03/2021 15:44:28	Plataforma	Alcanzado tiempo definido para timeout (2TO3). Fin del proceso (2F) – Registro cuenta de empresa
08/03/2021 15:44:29	Funcional	Se presenta un mensaje de error al cliente, indicando que no ha sido posible crear la cuenta.

Análisis de datos

Tras la ejecución del caso de estudio se pudo comprobar que el sustento teórico de la proposición de este trabajo de titulación ha podido ser materializado en una implementación específica para cubrir las necesidades de monitoreo del proceso de registro de clientes para una empresa local. El lenguaje SML ha sido aplicado para expresar los requerimientos de monitoreo planteados mediante un modelo gráfico. De acuerdo al objetivo de este caso de estudio, se ha podido validar que efectivamente el lenguaje tiene la capacidad de expresión suficiente para describir diversos escenarios de monitoreo, y por medio de una transformación modelo a texto se pueden desplegar servicios de monitoreo mediante una plataforma en la nube.

De acuerdo a lo definido en el conjunto de datos para la validación inicial de este caso de estudio, para analizar el esfuerzo necesario que debe realizar la empresa para implementar un sistema de

monitoreo de acuerdo a las necesidades, se ha solicitado al equipo de desarrollo de la empresa un detalle de todas las actividades necesarias y el esfuerzo aproximado en horas-hombre para especificar, diseñar, desarrollar y desplegar un sistema de monitoreo para cubrir los escenarios y flujos del proceso de registro de los clientes de la empresa por medio del sitio web, el mismo que se muestra en la tabla 15.

Tabla 15-Esfuerzo en proceso de implementación tradicional de sistema de monitoreo

Actividad	Personal requerido	Tiempo (horas hombre)
Especificación de requerimientos de monitoreo	Stakeholder, Ingeniero de software	16 h
Diseño arquitectura sistema de monitoreo	Arquitecto de TI, Ingeniero de software	4 h
Diseño diagrama de clases	Ingeniero de software	5 h
Diseño diagramas de secuencia	Ingeniero de software	6 h
Desarrollo servicios web central de monitoreo	Ingeniero de desarrollo senior	56 h
Desarrollo mecanismos de control en servicios web	Ingeniero de desarrollo senior	32 h
Desarrollo consumo servicios web - Aplicación de registro	Ingeniero de desarrollo senior	15 h
Desarrollo consumo servicios web - Servicios web Novelty	Ingeniero de desarrollo senior	20 h
Desarrollo consumo servicios web - Servicios web pagos	Ingeniero de desarrollo senior	5 h
Desarrollo servicios web respuesta a eventos - Aplicación de Registro	Ingeniero de desarrollo senior	48 h
Testing unitario - Comunicaciones Aplicación de registro – Central	Ingeniero de certificación estándar	4 h
Testing unitario - Comunicaciones Servicios web Novelty – Central	Ingeniero de certificación estándar	5 h
Testing unitario - Comunicaciones Servicios web pagos – Central	Ingeniero de certificación estándar	3 h
Testing integral – Escenario Registro, validación de datos y suscripción completa	Ingeniero de certificación senior	8 h
Testing integral – Escenario Registro sin suscripción	Ingeniero de certificación senior	6 h
Testing integral – Escenario Error inesperado en pasarela de pago	Ingeniero de certificación senior	6 h
Testing integral – Escenario Error en servicios web para creación de cuenta	Ingeniero de certificación senior	4 h
Despliegue de servicios en la nube y en los componentes cliente	Ingeniero de producción senior	8 h

Por otro lado, de la observación realizada durante el proceso de diseño con el personal designado por la empresa se han registrado las actividades y el esfuerzo necesario para elaborar el modelo de monitoreo utilizando el lenguaje SML, transformación del modelo a código fuente, configuración de componentes a monitorear y despliegue de servicios de monitoreo en la plataforma en la nube, como se muestra en la tabla 16.

Tabla 16-Esfuerzo en proceso de implementación con SML

Actividad	Personal requerido	Tiempo (horas hombre)
Especificación de requerimientos de monitoreo	Stakeholder	9 h
Elaboración del modelo de monitoreo con SML	Ingeniero de software	10 h
Transformación de modelo a texto (generación de código fuente)	Ingeniero de software	1 h
Creación de plataforma web e integración de código fuente servicios de monitoreo	Ingeniero de desarrollo senior	5 h
Integración código fuente en componente Aplicación de registro	Ingeniero de desarrollo senior	5 h
Integración código fuente en componente Servicio web Novelty	Ingeniero de desarrollo senior	5 h
Integración código fuente en componente Servicios web pagos	Ingeniero de desarrollo senior	5 h
Testing unitario - Comunicaciones Aplicación de registro - Central	Ingeniero de certificación estándar	4 h
Testing unitario - Comunicaciones Servicios web Novelty - Central	Ingeniero de certificación estándar	5 h
Testing unitario - Comunicaciones Servicios web pagos – Central	Ingeniero de certificación estándar	3 h
Testing integral – Escenario Registro, validación de datos y suscripción completa	Ingeniero de certificación senior	6 h
Testing integral – Escenario Registro sin suscripción	Ingeniero de certificación senior	4 h
Testing integral – Escenario Error inesperado en pasarela de pago	Ingeniero de certificación senior	6 h
Testing integral – Escenario Error en servicios web para creación de cuenta	Ingeniero de certificación senior	4 h
Despliegue de servicios en la nube y en los componentes cliente	Ingeniero de producción senior	12 h

De forma similar, para poder analizar el esfuerzo operativo que se realiza en la empresa para verificar cada evento de monitoreo en los diferentes casos planteados para el proceso de registro de clientes, se ha solicitado a la empresa un detalle de actividades que son llevadas a cabo durante

la verificación manual de cada caso reportado, junto con los tiempos necesarios para cada actividad, como se muestra en la tabla 17.

Tabla 17-Esfuerzo en proceso operativo de monitoreo tradicional

Actividad	Personal requerido	Tiempo (horas)
Verificar registro de cliente fallido por datos inválidos. (El cliente reporta un mensaje de error al soporte técnico. El personal encargado solicita los datos al cliente y procede a revisar los logs de los servicios web de Novelty, verificando que los datos se encuentran incorrectos)	Operador de soporte	0.5 h
Verificar creación correcta de cuenta de cliente en el sistema. (El cliente desea conocer si su cuenta ha sido creada correctamente. El personal encargado solicita los datos al cliente y procede a revisar las bases de datos de Novelty para verificar que la cuenta se ha creado correctamente)	Operador de soporte	0.25 h
Verificar proceso incompleto en registro de cliente sin suscripción. (El cliente indica que no puede acceder al sistema luego de registrarse. El personal encargado solicita los datos al cliente y procede a revisar las bases de datos de Novelty para verificar que no dispone de suscripción.)	Operador de soporte	0.5 h
Verificar error en ejecución de pago mediante pasarela. (El cliente informa de un error en la realización del pago. El personal encargado solicita los datos al cliente y procede a revisar los logs del componente Servicios web pagos y Aplicación de registro para validar la causa raíz del error reportado y reestablecer los servicios)	Operador de soporte	0.75 h
Verificar error en creación de cuentas de empresa. (El cliente informa de un error en la creación de cuenta de empresa. El personal encargado solicita los datos al cliente y procede a revisar los logs del componente Aplicación de registro y Servicios web novelty para validar la causa raíz del error reportado y reestablecer los servicios)	Operador de soporte	1.5 h

Finalmente, en base a los registros de tiempo encontrados en la plataforma de monitoreo tras la ejecución del caso de estudio se obtienen los tiempos necesarios para cada flujo de monitoreo en los diferentes escenarios, como se detalla en la tabla 18.

Tabla 18-Esfuerzo en proceso operativo de monitoreo con SML

Actividad	Personal requerido	Tiempo (horas)
Verificar registro de cliente fallido por datos inválidos. (El cliente reporta un mensaje de error al soporte técnico. El personal encargado solicita los datos al cliente y procede a	Operador de soporte	0.25 h

revisar los registros de la plataforma de monitoreo, verificando que los datos se encuentran incorrectos)		
Verificar creación correcta de cuenta de cliente en el sistema. (La plataforma de monitoreo despacha un evento cuando se ha realizado un registro de cliente siguiendo todo el flujo completo. El tratamiento del evento queda a discreción de la implementación que se realice, siendo en este caso la notificación por correo electrónico al cliente de creación correcta de cuenta)	Operador de soporte	0 h
Verificar proceso incompleto en registro de cliente sin suscripción. (La plataforma de monitoreo despacha un evento cuando se ha realizado un registro de cliente sin suscripción. El tratamiento del evento queda a discreción de la implementación que se realice, siendo en este caso la notificación por correo electrónico al personal designado de ventas para que proporcione seguimiento al cliente)	Operador de soporte	0.15 h
Verificar error en ejecución de pago mediante pasarela. (La plataforma de monitoreo despacha un evento cuando se ha presentado un error en la ejecución de pago mediante pasarela. El tratamiento del evento queda a discreción de la implementación que se realice, siendo en este caso la notificación por correo electrónico al personal designado de soporte técnico para reestablecer los servicios e informar oportunamente a los clientes)	Operador de soporte	0.5 h
Verificar error en creación de cuentas de empresa. (La plataforma de monitoreo despacha un evento cuando se ha presentado un error en la creación de cuenta de empresa. El tratamiento del evento queda a discreción de la implementación que se realice, siendo en este caso la notificación por correo electrónico al personal designado de soporte técnico para reestablecer los servicios e informar oportunamente a los clientes)	Operador de soporte	0.5 h

Resultados

Luego del análisis realizado al caso de estudio aplicado se presentan a continuación los resultados, beneficios y debilidades encontradas en la implementación y ejecución del sistema de monitoreo basado en el lenguaje SML para el proceso de registro de clientes de la empresa.

La principal contribución que muestra el lenguaje SML es la capacidad de representar las necesidades de monitoreo de sistemas planteadas a nivel de diseño, con un lenguaje enriquecido con la capacidad de expresión general para múltiples soluciones de monitoreo. Este enfoque en la fase de diseño desprende beneficios asociados al esfuerzo o tiempo necesario en el proceso de implementación de una solución de monitoreo, así como también en el proceso operativo de soporte de primer nivel a los usuarios de los sistemas.

La aplicación de SML muestra considerables mejoras con respecto al tiempo necesario en las fases de implementación. Como primer resultado se evidencia que el proceso de implementación ha sufrido una redistribución de esfuerzos en los tiempos utilizados para cada actividad. En las figuras 26 y 27 se observa la diferencia que existe entre el proceso de implementación tradicional versus la implementación propuesta en este trabajo. Con SML la actividad de desarrollo es reducida considerablemente, lo cual repercute directamente en la distribución porcentual de esfuerzos para las actividades de especificación de requerimientos, diseño, *testing* y despliegue.

Esta redistribución de esfuerzos se traduce directamente en una optimización de tiempo, como se muestra en la figura 28. Se tiene un ahorro considerable de tiempo en las fases de especificación de requerimientos, diseño, *testing* y principalmente desarrollo, con la reducción más notable de tiempo. Es importante mencionar que la actividad de despliegue de servicios ha mostrado una mayor duración en la implementación con SML, debido a la carga de servicios y comunicaciones que deben ser provistas en cada uno de los componentes y la plataforma de monitoreo.

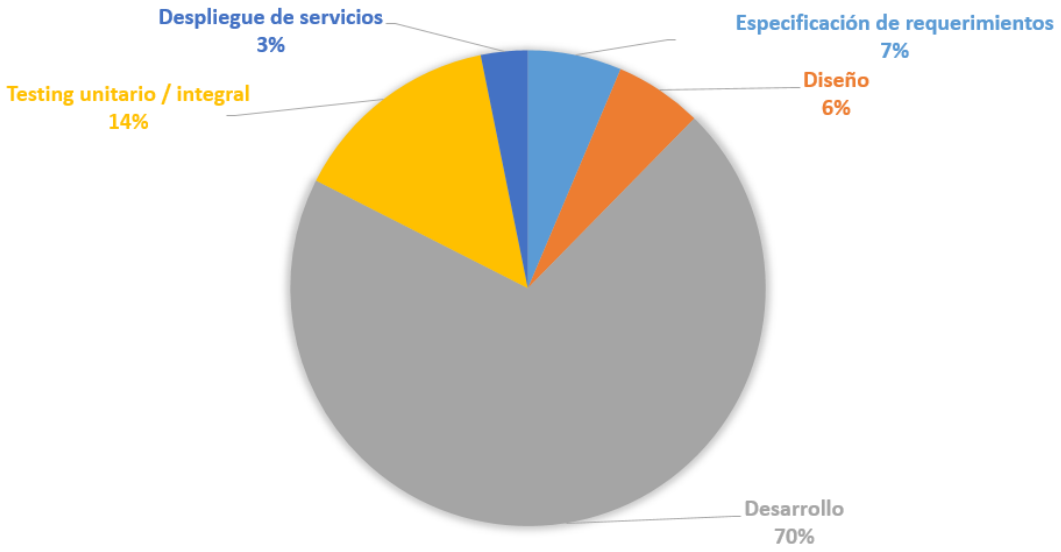

Figura 26-Distribución de actividades en implementación tradicional

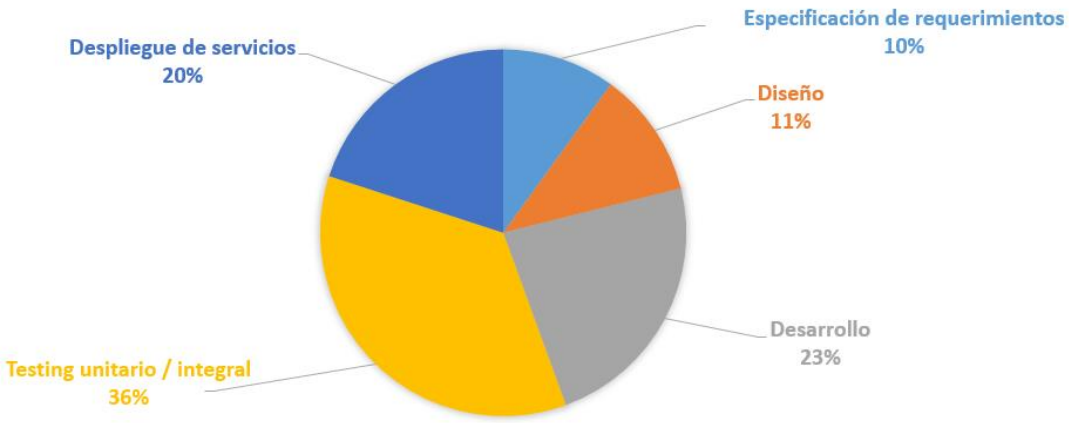

Figura 27-Distribución de actividades en implementación SML

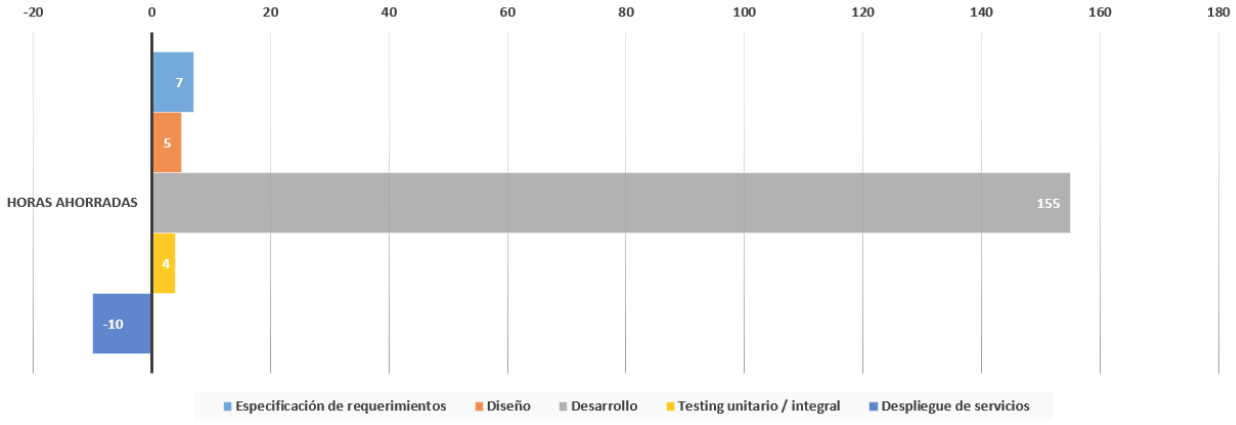

Figura 28-Diferencia en horas entre proceso tradicional vs SML

Al analizar las fases de implementación en conjunto, se observa que con SML la implementación del sistema de monitoreo es mucho más rápida que siguiendo el modelo tradicional. En la figura 29 se observa la comparativa de esfuerzo-tiempo necesario para implementar el sistema de monitoreo bajo los dos enfoques.

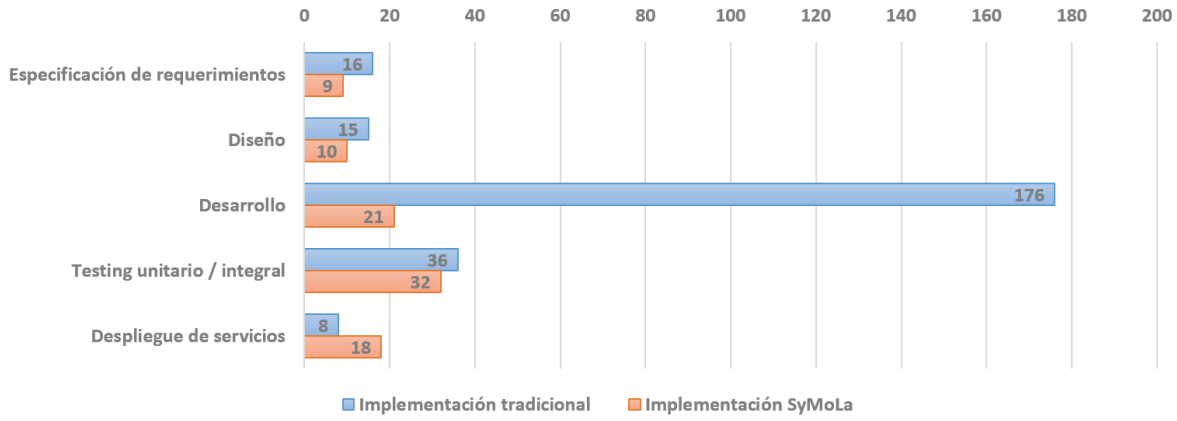

Figura 29-Comparativa de esfuerzo-tiempo en implementación de sistemas de monitoreo

Finalmente, el análisis realizado al proceso que realiza el personal de soporte técnico en la empresa respecto al esfuerzo necesario para las actividades de soporte a los clientes ha mostrado que la implementación de SML en la empresa puede reducir considerablemente el tiempo de respuesta al cliente o en algunos casos prescindir del soporte. La figura 30 muestra la comparativa de tiempo entre la ejecución manual del soporte técnico, y la utilización de la plataforma de monitoreo SML en apoyo a las actividades de soporte.

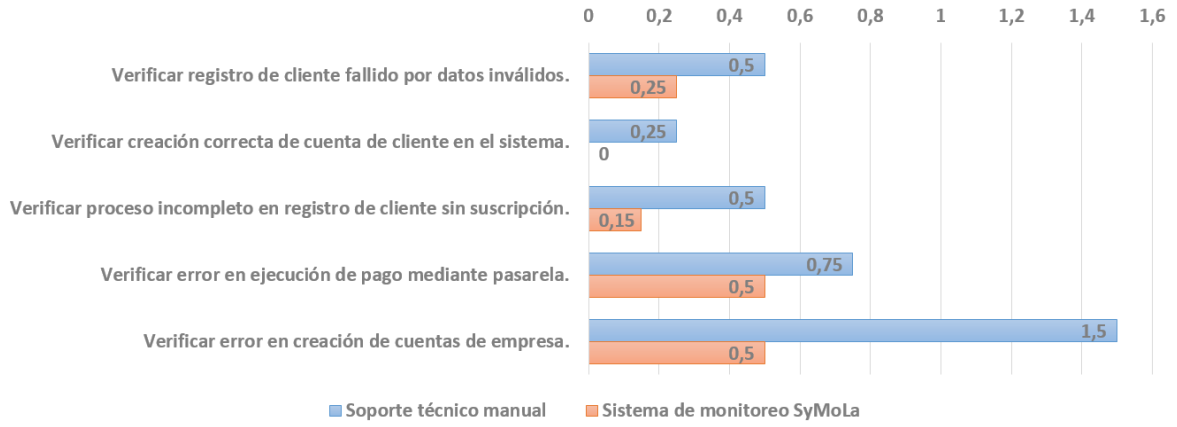

Figura 30-Comparativa de esfuerzo-tiempo en actividades de soporte técnico

Capítulo 5 – Conclusiones y trabajo futuro

Conclusiones

En el presente trabajo se ha propuesto un lenguaje de modelado para la especificación de sistemas de monitoreo, el cual presenta grandes ventajas por su capacidad de expresión de requerimientos de monitoreo en la fase de diseño, que serán reflejadas en la fase de implementación. Esto permite que el lenguaje SML sea utilizado por usuarios expertos y diseñadores que requieran plasmar el conocimiento de forma gráfica en un alto nivel de abstracción, sin entrar en detalles técnicos de

implementación. Se ha diseñado el lenguaje de modelado desde una perspectiva amplia y generalista, evitando la dependencia hacia tecnologías específicas o un dominio particular. Es por esto que el lenguaje SML puede ser utilizado en cualquier dominio de sistemas cuyos componentes tengan comunicación con la red, pudiendo incluso generar soluciones de monitoreo entre sistemas de sistemas en áreas como auditoría, entes reguladores, gobierno de TI, sistema financiero, internet de las cosas, comercio electrónico, entre otras.

Para permitir la fácil integración de cada uno de los componentes de los sistemas a monitorear, se ha propuesto un enfoque para despliegue de los servicios de monitoreo en un entorno en la nube, de manera que los componentes se comuniquen con la plataforma de monitoreo mediante protocolos estándar de comunicación. Esto genera un abanico de posibilidades para integración de servicios de monitoreo entre las plataformas y sistemas utilizados hoy en día, cerrando la brecha existente entre diferentes tecnologías, proveedores o empresas, al disponer de servicios de monitoreo diseñados por usuarios expertos y ajustados exactamente a las necesidades de la industria del software.

Por otro lado, el trabajar en un nivel superior de abstracción con el lenguaje de modelado SML junto con herramientas de transformación de modelo a texto, muestra grandes beneficios reduciendo el tiempo de implementación de las soluciones de monitoreo. Apoyarse en dichas herramientas permite automatizar en gran medida la generación de código fuente, con una evidente reducción de esfuerzo necesario en la fase de implementación, permitiendo a las empresas optimizar recursos mientras enfocan todo el conocimiento de los usuarios expertos para obtener diseños de mejor calidad.

En muchas empresas la información es un activo sumamente valioso. Al monitorear los sistemas se genera información de gran valor que permite la verificación, trazabilidad, transparencia, respuesta a incidentes en producción, entre otros. Incluso, esta información es insumo para encontrar correlaciones entre eventos, predecir fallas en los sistemas, ajustar comportamientos transaccionales o aplicar inteligencia de negocios como herramienta para la toma de decisiones. SML se presenta como una base sobre la cual se proyectan nuevas posibilidades de estudio y aplicaciones.

Trabajo futuro

Como trabajo futuro, se propone la aplicación de SML en la industria del software por medio de casos de estudio, que permitan analizar de forma cualitativa y cuantitativa los resultados obtenidos. Dependiendo del nivel de detalle de las transacciones y el tamaño de los sistemas a monitorear, se puede requerir gran cantidad de recursos de red, procesamiento o almacenamiento. Es por esto que se considera conveniente definir formatos estandarizados para la transferencia de datos entre los componentes y la plataforma, optimizando las comunicaciones y mejorando el rendimiento de respuesta.

Por otro lado, el rol de SML está centrado en la fase de diseño del software, por lo cual resulta importante analizar la integración con las siguientes fases del proceso, por medio de la transformación modelo a texto M2T para permitir la generación automatizada de plataformas de monitoreo.

Finalmente, en esta propuesta se ha utilizado un enfoque orientado a servicios en la nube, por los beneficios asociados en la integración de los sistemas a monitorear con la plataforma de monitoreo. Sin embargo, el lenguaje no está ligado a un modelo de despliegue específico, por lo cual resulta interesante analizar alternativas de despliegue diferentes al modelo Cliente-Servidor, como por ejemplo: modelos peer-to-peer.

Referencias

August, J., No, I., & Rathod, D. (2017). Performance evaluation of Restful web services and SOAP / WSDL web services. *International Journal of Advanced Research in Computer Science*, *8*(7). https://doi.org/http://dx.doi.org/10.26483/ijarcs.v8i7.4349

Bergert, M., Diedrich, C., Germany, M., Kiefer, J., & Bär, T. (2007). Automated PLC Software Generation Based on Standardized Digital Process Information Institute of Automation Technology (IFAT) Function and Production Modeling (GR / EPF). *IEEE*, 352–359. https://doi.org/1-4244-0826-1

Booch, G., & Rumbaugh, J. (1996). *The Unified Modeling Language for Object-Oriented Development* (pp. 1–35). pp. 1–35. Santa Clara: RATIONAL Software Corporation.

Cedillo, P., Gonzalez, J., Abrahao, S., & Insfran, E. (2016). A Monitoring Infrastructure for the Quality Assessment of Cloud Services. *ISD 2015 - Transforming Healthcare Through Information Systems*, 17–32. https://doi.org/10.1007/978-3-319-30133-4

Chaudhuri, S. R., Natarajan, S., Banerjee, A., & Choppella, V. (2019). Methodology to develop domain specific modeling languages. *DSM 2019 - Proceedings of the 17th ACM SIGPLAN International Workshop on Domain-Specific Modeling, Co-Located with SPLASH 2019*, 1–10. https://doi.org/10.1145/3358501.3361235

Chen, D., Maffei, A., & Ferreirar, J. (2015). Environment for the Management Environment for the Virtual Environment for the Management Environment for the Management Management Development Development Development and Development of Systems. *IFAC-PapersOnLine*, *48*(7), 29–36. https://doi.org/10.1016/j.ifacol.2015.06.469

Chen, D., Panfilenko, D. V., Khabbazi, M. R., & Sonntag, D. (2016). A Model-Based Approach to Qualified Process Automation for Anomaly Detection and Treatment. *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. https://doi.org/10.1109/ETFA.2016.7733731

Cicotti, G., Antonio, S. D., Cristaldi, R., & Sergio, A. (2013). How to Monitor QoS in Cloud Infrastructures : The QoSMONaaS Approach. *International Journal of Computational Science*

and Engineering, 253–262. https://doi.org/10.1504/IJCSE.2015.071359

Cuadrado, J. S., Cánovas Izquierdo, J. L., & Molina, J. G. (2014). Applying model-driven engineering in small software enterprises. *Science of Computer Programming*, *89*(PART B), 176–198. https://doi.org/10.1016/j.scico.2013.04.007

Davison, A. P., Hines, M. L., & Muller, E. (2009). Trends in programming languages for neuroscience simulations. *Frontiers in Neuroscience*, *3*(DEC), 374–380. https://doi.org/10.3389/neuro.01.036.2009

Diprose, J., Macdonald, B., Hosking, J., Plimmer, B., Diprose, J., Macdonald, B., & Hosking, J. (2016). Author ' s Accepted Manuscript Designing an API at an appropriate abstraction level for programming social robot applications Reference : *Journal of Visual Language and Computing*. https://doi.org/10.1016/j.jvlc.2016.07.005

Divoux, T., Rondeau, E., & Lepage, F. (1997). Using the EXPRESS language as a reference interface to de ® ne MMS communication. *Journal of Intelligent Manufacturing - Springer*. https://doi.org/10.1023/A:1018544418379

Dotcom-Monitor. (2020). Casos de uso de monitoreo web. Retrieved from https://www.dotcom-monitor.com/wiki/es/knowledge-base/casos-de-uso-de-monitoreo-web/

Efendioglu, N., & Woitsch, R. (2017). A Modelling Method for Digital Service Design and Intellectual Property Management Towards Industry 4 . 0 : CAxMan Case. *Springer International Publishing*, 153–163. https://doi.org/10.1007/978-3-319-61240-9

Garcia, M. V, Irisarri, E., Perez, F., Estevez, E., & Marcos, M. (2016). OPC-UA communications integration using a CPPS architecture BT -. *IEEE Ecuador Technical Chapters Meeting, ETCM 2016.* https://doi.org/10.1109/ETCM.2016.7750838

Givehchi, O., Landsdorf, K., Simoens, P., & Colombo, A. W. (2017). *Interoperability for industrial cyber-physical systems : an approach for legacy systems*. *3203*(c). https://doi.org/10.1109/TII.2017.2740434

Gosling, J., & Buckley, A. (2011). *The Java Language Specification* (Java SE 7). Retrieved from docs.oracle.com

Gray, J., Bapty, T., Neema, S., Schmidt, D. C., & Al, B. (2003). An Approach for Supporting Aspect-Oriented Domain Modeling. *International Conference on Generative Programming and Component Engineering*, *2830*, 151–168.

Gutierrez, J. M., & Holgado, J. A. (2017). iMMAS an Industrial Meta-Model for Automation System Using OPC UA. *Elektronika IR Elektrotechnika*, *23*, 3–11. https://doi.org//10.5755/j01.eie.23.3.18324

Harcuba, O., & Vrba, P. (2015). Ontologies for Flexible Production Systems. *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. https://doi.org/10.1109/ETFA.2015.7301482

Hildebrandt, C., Glawe, M., Müller, A. W., & Fay, A. (2017). Reasoning on Engineering Knowledge :

Applications and Desired Features. *European Semantic Web Conference*, *10250*, 65–78. https://doi.org/10.1007/978-3-319-58451-5

Holub, V., Parsons, T., Sullivan, P. O., Svt, W., & Murphy, J. (2009). Run-Time Correlation Engine for System Monitoring and Testing Categories and Subject Descriptors. *ICAC-INDST '09: Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*, 9–17. https://doi.org/10.1145/1555312.1555317

Hutchison, D., & Mitchell, J. C. (2011). On the Move to. *Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011 Hersonissos*. Crete, Greece. https://doi.org/10.1007/978-3-642-25106-1

Inzinger, C., Hummer, W., Satzger, B., & Leitner, P. (2014). Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems. *Wiley Online Library*, (January), 805–822. https://doi.org/10.1002/spe.2254

Joyce, J., Lomow, G., Slind, K., & Unger, B. (1987). Monitoring Distributed Systems. *ACM Transactions on Computer Systems (TOCS)*, *5*(2), 121–150. https://doi.org/10.1145/13677.22723

Ka, B. (2013). Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems. *CIRP Annals - Manufacturing Technology*, *62*, 443–446. https://doi.org/10.1016/j.cirp.2013.03.045

Karsai, G., Rumpe, B., Schindler, M., & Völkel, S. (2009). Design Guidelines for Domain Specific Languages. *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM' 09)*, (October).

Kouhen, A. El, Dumoulin, C., Gérard, S., Boulet, P., Kouhen, A. El, Dumoulin, C., … Boulet, P. (2012). Evaluation of Modeling Tools Adaptation. *HAL Archives-Ouvertes*. https://doi.org/hal-00706701

Kovalenko, O., Wimmer, M., Sabou, M., Lüder, A., Ekaputra, F. J., & Biffl, S. (2015). Modeling AutomationML : Semantic Web Technologies vs . Model-Driven Engineering. *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2–5. https://doi.org/10.1109/ETFA.2015.7301643

Lauriac, N. (2016). *Diseño e implementación de un sistema de monitoreo*. Lausanne: Terre des hommes. Retrieved from tdh.ch

Lütjen, M., & Rippel, D. (2015). GRAMOSA framework for graphical modelling and simulation-based analysis of complex production processes. *The International Journal of Advanced Manufacturing Technology*, *81*. https://doi.org/10.1007/s00170-015-7037-y

Mazak, A., & Huemer, C. (2015). A Standards Framework for Value Networks in the Context of Industry 4 . 0. *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 1342–1346. https://doi.org/10.1109/IEEM.2015.7385866

Mechs, S., Grimm, S., Beyer, D., & Lamparter, S. (2013). Evaluation of prediction accuracy for energy-efficient switching of automation facilities. *IECON Proceedings (Industrial Electronics*

Conference), 6928–6933. https://doi.org/10.1109/IECON.2013.6700281

Mernik, M. (2017). Domain-Specific Languages : A Systematic. *International Conference on Current Trends in Theory and Practice of Informatics*, 464–472. https://doi.org/10.1007/978-3-319-51963-0

Meyer, B. (2013). *Touch of class*. New York: Springer Science+Business Media. Retrieved from www.springer.com

Mockford, K. (2004). Web Services architecture. *BT Technology Journal*, *22*(1), 19–26. Retrieved from https://link.springer.com/article/10.1023/B:BTTJ.0000015492.03732.a6

Negri, E., Fumagalli, L., Garetti, M., & Tanca, L. (2015). Computers in Industry Requirements and languages for the semantic representation of manufacturing systems. *Computers in Industry*. https://doi.org/10.1016/j.compind.2015.10.009

Nielsen, H. F., & Thatte, S. R. (2014). Simple Object Access Protocol (SOAP) 1 . 1. Retrieved from http://www.w3.org

ObeoSoft. (2021). OBEO. Retrieved May 2, 2021, from https://www.obeosoft.com/en/products

Olaru, M. A. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective. *7th International Conference Interdisciplinarity in Engineering (INTER-ENG 2013) Advantages*, *12*, 529–534. https://doi.org/10.1016/j.protcy.2013.12.525

Paige, R. F., Ostroff, J. S., & Brooke, P. J. (2000). Principles for modeling language design. *Information and Software Technology*, *42*(10), 665–675. https://doi.org/10.1016/S0950-5849(00)00109-9

Petrasch, R., & Hentschke, R. (2016). Process Modeling for Industry 4 . 0 Applications. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, (Cc). https://doi.org/10.1109/JCSSE.2016.7748885

Rodrigues Da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems and Structures*, *43*, 139–155. https://doi.org/10.1016/j.cl.2015.06.001

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Sadigh, B. L., Unver, H. O., Nikghadam, S., Ozbayoglu, A. M., & Kilic, S. E. (2016). An ontology-based multi-agent virtual enterprise system (OMAVE): part 1 : domain modelling and rule management. *International Journal of Computer Integrated Manufacturing*, *3052*(February), 0–24. https://doi.org/10.1080/0951192X.2016.1145811

Schubert, D., Gerking, C., & Heinzemann, C. (2016). Towards Safe Execution of Reconfigurations in Cyber-Physical Systems. *19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*. https://doi.org/10.1109/CBSE.2016.10

Sendall, S., & Kozaczynsk, W. (2003). Model Transformation : The Heart and Soul of. *IEEE Software*,

20. https://doi.org/10.1109/MS.2003.1231150

Shaw, M. (1990). Toward higher-level abstractions for software systems. *Third International Symposium on Knowledge Engineering*, *5*(May 1989), 119–128. https://doi.org/10.1016/0169-023X(90)90008-2

Sousa, G. C. M., Costa, F. M., Clarke, P. J., & Allen, A. A. (2012). Model-driven development of DSML execution engines. *Proceedings of the 7th Workshop on Models@run.Time, MRT 2012 - Being Part of the ACM/IEEE 15th International Conference on Model Driven Engineering Languages and Systems, MODELS 2012*, 10–15. https://doi.org/10.1145/2422518.2422521

Steven, K., & Tolvanen, J.-P. (2008). *Domain-Specific Modeling Enabling full code generation* (J. W. & Sons, Ed.). IEEE Computer Society. Retrieved from www.dsmbook.com

Strang, D., & Anderl, R. (2014). Assembly process driven component data model in cyber-physical production systems. *World Congress on Engineering and Computer Science 2014 Vol*, *2*(November), 947–952. https://doi.org/2078-0966

Thramboulidis, K., & Christoulakis, F. (2016). Computers in Industry UML4IoT — A UML-based approach to exploit IoT in cyber-physical manufacturing systems. *Computers in Industry*. https://doi.org/10.1016/j.compind.2016.05.010

Walch, M. (2017). Knowledge-driven Enrichment of Cyber-physical Systems for Industrial Applications Using the KbR Modelling Approach. *IEEE International Conference on Agents (ICA)*, 84–89. https://doi.org/10.1109/AGENTS.2017.8015307

Wortmann, A., Barais, O., Combemale, B., & Wimmer, M. (2020). Modeling languages in Industry 4.0: an extended systematic mapping study. *Software and Systems Modeling*, *19*(1), 67–94. https://doi.org/10.1007/s10270-019-00757-6

Zhu, C. (2009). The Socket Programming and Software Design for Communication Based on Client / Server. *Pacific-Asia Conference on Circuits,Communications and System*, 775–777. https://doi.org/10.1109/PACCS.2009.89