Automatic Code Generation of Data Visualization for Structural Health Monitoring

Braulio Quiero and Gonzalo Rojas

***Abstract*—Structural Health Monitoring (SHM) aims at detecting, localizing, and characterizing damages in civil, mechanical, and aerospatial structures, which are hardly detectable in visual inspections. The collection, analysis, and visualization of data captured by sensors installed on these structures can be strongly supported by modern techniques of Data Science. In particular, the visualization of these data provides valuable help to experts on structural health and decision makers on preventive and corrective maintenance. Unfortunately, existing systems of data visualization still demand those stakeholders for a high level of software programming skills to take full advantage of visual and interactive exploration of data that sensors capture and output.**

This work introduces a model-driven approach to develop data visualization in the domain of structural health monitoring, in particular, of bridges. This approach is based on the definition of a Domain Specific Language (DSL) that describes the main concepts of an infrastructure of sensors typically used in SHM, along with common graphics and visual alternatives of data visualization. This DSL is instantiated by a modeling language, composed of a metamodel, a visual representation of concepts, and a set of model-to-text transformation rules. In this way, non-programmers can implement their own data visualization from a graphical and intuitive design, by automatically generating the corresponding code. This approach was implemented in a modeling and code-generation tool, called Vis4bridge, whose usability and output were successfully evaluated through the development of tasks and case studies.

***Index Terms*—Structural Health Monitoring, Model Driven Software Development, Data Visualization, Domain Specific Languages.**

I. Introducción

El Monitoreo de Salud Estructural (SHM, del inglés Structural Health Monitoring) [1] es una disciplina orientada a la detección, localización [2] y caracterización de aquellos daños en estructuras de ingeniería civil, mecánica y aeroespacial, que son difícilmente detectables mediante inspección visual. Para ello, se analizan datos provenientes de múltiples sensores instalados en la estructura a monitorear o en elementos que interactúen con ella.

El análisis de grandes volúmenes de datos, como es el caso de las mediciones de sensores de sistemas SHM, se ve ampliamente favorecido por técnicas de visualización de datos. Sin embargo, la creación e implementación de estas visualizaciones es altamente desafiante para expertos en el dominio con un bajo conocimiento de desarrollo de software [3]. Aun cuando ha aumentado la oferta de bibliotecas y entornos de desarrollo orientados a la visualización, el nivel de conocimiento técnico requerido sigue siendo alto. En monitoreo estructural, esta brecha atenta contra la autonomía de los expertos para explorar visualmente sus datos de acuerdo a los requerimientos de su dominio.

El presente trabajo postula que la utilización de un enfoque de Desarrollo de Software Dirigido por Modelos permitirá acortar la brecha de conocimiento técnico descrita, simplificando la implementación de visualizaciones de datos por parte de expertos en monitoreo estructural. Prescindiendo de conocimientos avanzados de programación, estos expertos del dominio podrán generar sus propias alternativas de visualización, con tiempos de desarrollo inferiores al desarrollo tradicional.

Con este objetivo, se presenta a continuación un Lenguaje Específico de Dominio para visualizaciones de datos masivos. Este lenguaje (DSL, Domain Specific Language) se expresa como un lenguaje de modelado, cuya sintaxis abstracta corresponde a un metamodelo que describe un conjunto extensible de alternativas de visualización (gráficos de línea, histogramas, diagramas de caja, entre otros), y una infraestructura de sensores (como acelerómetros, inclinómetros, strain gauges), también extensible, cuyas mediciones serán visualizadas. La sintaxis concreta del lenguaje introducido comprende la definición de una representación gráfica, que permitirá a expertos del dominio describir sus requerimientos de visualización de forma fácil e intuitiva, abstrayéndoles de conocimientos de programación. Finalmente, la semántica del lenguaje de modelado se expresa mediante un conjunto de transformaciones de modelo a texto, que mapean estructuras del metamodelo a su correspondiente implementación en código fuente.

La implementación resultante corresponde a un sistema web correspondiente a un panel de visualización, el que está compuesto por todas las alternativas de visualización modeladas. Estas alternativas, gráficas o textuales, despliegan los datos de mediciones de sensores para su análisis por parte de expertos en monitoreo estructural de puentes. La propuesta se enfoca en datos que ya han sido recibidos desde una infraestructura de sensores instalada en un puente bajo monitoreo, y que tengan una antigüedad a definir por parte de los actores interesados.

Los tres componentes de este DSL fueron implementados en una herramienta de modelado y generación automática de código. Su desarrollo responde a requerimientos de visualización y análisis de datos, recientes e históricos, planteadas por Ingenieros Civiles expertos en monitoreo estructural. Se analizaron los usos más frecuentes de distintas alternativas de visualización provistas por bibliotecas de lenguajes de programación, junto con visualizaciones provistas en herramientas software de monitoreo comerciales. Se analizaron, además,

Braulio Quiero, Department of Computer Science, University of Concepcion, Chile e-mail:bquiero@inf.udec.cl.

Gonzalo Rojas, Department of Computer Science, University of Concepcion, Chile e-mail:gonzalorojas@inf.udec.cl.

propuestas de desarrollo dirigidas por modelos relacionadas con visualizaciones de datos e interfaces de usuario.

La propuesta se evaluó mediante estudio de casos, donde expertos en monitoreo generaron distintas alternativas de visualización, sometiéndoles a cuestionarios de usabilidad de la herramienta, con auspiciosos resultados.

El resto del documento se organiza como sigue: la Sección II describe trabajos relacionados en visualización para monitoreo estructural y en el desarrollo dirigido por modelos de visualizaciones de datos. La Sección III presenta el DSL descrito y los tres componentes del lenguaje de modelado que lo implementa. La Sección IV describe la arquitectura y la implementación de la herramienta de modelado y generación de código basada en el DSL. La Sección V describe el proceso y resultados de la evaluación aplicada. Finalmente, la Sección VI presenta conclusiones y trabajos futuros.

II. Trabajos Relacionados

Existe una creciente actividad de desarrollo de sistemas software para soportar el monitoreo de salud estructural, basado en datos masivos provenientes de sensores. Sin embargo, la visualización de datos asociados a monitoreo cuenta con escasas propuestas en términos de proceso y producto software final. El desarrollo de visualizaciones a disposición de usuarios, es decir, no orientadas sólo a depurar procesos de análisis y predicción, se ha concentrado principalmente en herramientas comerciales. La intensa actividad investigadora en procesos de captura, almacenamiento y análisis de datos en el contexto del monitoreo estructural, no se traspasa a la visualización de dichos datos por medio de gráficos y alternativas textuales.

Entre aquellas propuestas que ofrecen visualizaciones gráficas de dichos datos, destacan principalmente aquellos sistemas que ofrecen monitoreo en tiempo real. Por ejemplo, Masri et al. [4] presentan un sistema de monitoreo estructural de puentes en tiempo real, que permite visualizar datos de grupos de sensores seleccionados por el usuario, a través de mecanismos de publicación y suscripción. El sistema permite visualizar dichos datos mediante transformadas rápidas de Fourier (FFT), media cuadrática (RMS) y la correlación entre datos de dos sensores. Todas estas representaciones se realizan exclusivamente por medio de gráficos de línea (*line charts*), diferenciando las señales de acelerómetros triaxiales por color. Este trabajo es un aporte a la visualización de datos de monitoreo en tiempo real, pero no considera visualizaciones de datos ya recibidos y almacenados. El sistema de monitoreo presentado en [5] presenta gráficos de línea de mediciones en tiempo real de múltiples sensores, y animaciones de la vibración del puente monitoreado ante eventos específicos. La selección de eventos específicos a consultar es una interesante contribución al análisis de datos históricos, aun cuando el trabajo no precisa la antigüedad de los datos posible de consultar. En [6], se muestra un sistema de monitoreo con interfaz de alertas sobre el estado de las vigas de un puente, y gráficos de línea para visualizar, en tiempo real, múltiples señales de galgas extensométricas (*strain gauges*) instaladas en distintos niveles de la estructura. Este sistema muestra la necesidad de considerar distintos tipos de sensores, además de los acelerómetros que son comunes a las implementaciones mencionadas. Otras alternativas de presentación, distintas a gráficos de línea, se observan en el sistema de monitoreo en tiempo real de equipamiento eléctrico industrial presentado en [7]. En este caso, se utilizan tarjetas de datos (*cards*, valor de un dato relevante) con el valor de corriente actual y día y hora del valor, un registro tabular de estados de operación, valores de corriente y su fecha, y metáforas gráficas que informan sobre el estado global del equipamiento (normal, parado, alerta).

Con respecto a la generación automática de visualizaciones de datos, en [8] se presenta un lenguaje específico de dominio para diseñar distintas variantes de implementación para gráficos comunes (como gráficos circulares (*pie charts*) o de barras (*bar charts*)), antes de su implementación final. El lenguaje es textual, careciendo de una sintaxis gráfica que facilite su utilización por parte de quienes no son especialistas en programación. Sin embargo, su alta expresividad permite describir múltiples variantes de gráficas comunes, en un alto nivel de abstracción. En [3], se presenta un DSL orientado a la generación de visualizaciones de datos geoespaciales masivos, en una arquitectura pipeline que preprocesa datos crudos y los formatea, llamando a bibliotecas gráficas de Google Maps e integrando sus resultados en implementaciones HTML y Javascript. Aunque no presenta conceptos ni ejemplos de la descripción a alto nivel de las visualizaciones resultantes, este trabajo es una valiosa contribución a la integración de datos masivos con su despliegue gráfico. Por su parte, Brambilla y Umuhoza [9] presentan un DSL para la especificación de interfaces gráficas que faciliten la interacción con dispositivos de *Internet of Things* (IoT). Además, el trabajo presenta un panel (*dashboard*) con gráficas de visualización en tiempo real, con tarjetas de datos, medidores (*gauges*) y gráficos de línea. Sin embargo, el DSL presentado se enfoca en la gestión e interacción de dispositivos IoT, pero no se distinguen los conceptos que describen el panel y sus gráficos.

De esta revisión, podemos concluir que la visualización de datos en monitoreo estructural es considerada principalmente en un régimen de monitoreo en tiempo real. Para ello, se reconoce el uso del gráfico de línea como una alternativa adecuada para visualizar datos de forma cronológica (series de tiempo). En menor medida, se usan otras alternativas gráficas, como tarjetas de datos, tablas y metáforas gráficas de la estructura o equipamiento monitoreado. Estas alternativas se ajustan tanto al requerimiento de visualización en tiempo real, como al de la naturaleza de los sensores utilizados, principalmente acelerómetros triaxiales y extensiómetros. Sin embargo, la utilización de gráficas que se adecuen a objetivos de análisis de datos para datos ya recibidos y almacenados, como histogramas para análisis de distribución, *diagramas de caja* para dispersión con respecto a la mediana, *OHLC (open-high-low-close)* para tendencias, o gráficos de dispersión (*scatterplot*) para correlación, no se incluyen en los sistemas analizados.

La Tabla I muestra una comparativa de las propuestas revisadas en visualización para monitoreo estructural, que ilustra la poca variedad de alternativas de visualización ofrecidas y la casi exclusiva atención a visualizaciones en tiempo real.

TABLA I
COMPARATIVA DE TRABAJOS EN VISUALIZACIÓN PARA SISTEMAS DE MONITOREO ESTRUCTURAL FRENTE A LA PRESENTE PROPUESTA

Trabajo	Gráficos soportados	Almacenados / Tiempo Real	Antigüedad de datos
[4]	Gráficos de línea	Tiempo real	–
[5]	Gráficos de línea	Almacenados y Tiempo real	no precisada
[6]	Gráficos de Línea	Tiempo real	–
[7]	Tarjetas, tablas, metáforas gráficas	Tiempo real	–
Propuesta	Gráficos de línea, de área, histogramas, dispersión, tablas, tarjetas	Almacenados	Variable (evaluada para 14 días)

TABLA II
COMPARATIVA DE TRABAJOS DESTACADOS EN DESARROLLO DIRIGIDO POR MODELOS DE VISUALIZACIONES PARA SISTEMAS SHM

Trabajo	Lenguaje	Modelado de Visualizaciones	Modelado de Sensores	Integración Visualización Sensor
[8]	Textual	Sí	No	No
[3]	Gráfico	No	No	No
[9]	Gráfico	No	Sí	No
Propuesta	Gráfico	Sí	Sí	Sí

La propuesta que se presenta tiene como objetivo abarcar una amplia y creciente variedad de visualizaciones para datos almacenados, con una antigüedad a definir.

Por otra parte, en las propuestas revisadas de lenguajes específicos de dominio relacionados con la visualización de datos, se distinguen aportes en la descripción a alto nivel de múltiples alternativas gráficas [8], en la especificación de mecanismos para transformar datos crudos en inputs adecuados para su integración con bibliotecas gráficas [3], y en la gestión de dispositivos que capturan los datos a visualizar [9]. Dichos lenguajes, sin embargo, no consideran la correspondencia entre este tipo de dispositivos (por ejemplo, sensores de monitoreo o equipos IoT) con la representación gráfica de los datos que capturan, como tampoco una sintaxis gráfica que facilite la elaboración de modelos por parte de profesionales no expertos en programación.

La Tabla II compara los objetivos de esta propuesta con los trabajos revisados sobre aproximaciones dirigidas por modelos y DSL para visualización de datos o sensores de monitoreo. Una propuesta carece de lenguaje gráfico, lo que dificulta su uso por parte de inexpertos en programación. Sin embargo, esta es la única propuesta que provee modelado de visualizaciones. Sólo una presenta un modelado de sensores, pero ninguna integra, a nivel de modelado, datos de sensores y las visualizaciones de sus mediciones. Así, otro de los objetivos de la presente propuesta es el dotar de un lenguaje de modelado gráfico para visualizaciones, sensores y las asociaciones entre ambos grupos.

En conclusión, en sistemas software de apoyo al monitoreo estructural, se advierte una carencia en propuestas que aborden la especificación de visualizaciones de datos adecuadas a la exploración de datos más antiguos que los datos instantáneos desplegados en la revisión en tiempo real. Esto dificulta la realización de análisis más complejos, como la distribución de mediciones en el tiempo, dispersiones, o estadísticos significativos. Por otra parte, la generación automática de dichas visualizaciones no está adecuadamente soportada por lenguajes gráficos que permitan a profesionales del dominio, sin conocimientos técnicos de programación, participar activamente en la implementación de sus requerimientos de visualización.

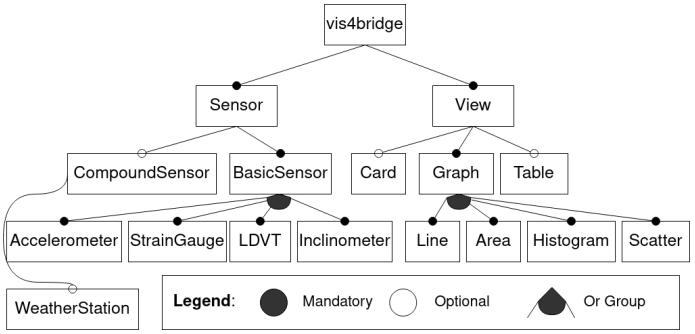


Fig. 1. Modelo de características para la definición del DSL.

III. DESCRIPCIÓN DE LA PROPUESTA

Este trabajo presenta un marco de desarrollo dirigido por modelos, basado en un Lenguaje Específico de Dominio (DSL) para visualizaciones de datos obtenidos a partir de sensores, en el contexto del Monitoreo de Salud Estructural de Puentes. En los Sistemas de Monitoreo de Salud Estructural, múltiples sensores realizan mediciones a altas frecuencias, para apoyar análisis en tiempo real, reciente, histórico y predictivo de alta precisión y confiabilidad. Esta propuesta se orienta al desarrollo de alternativas de visualización frecuentemente utilizadas para el análisis de datos obtenidos a partir de sensores.

Adoptando la definición introducida en [10], que describe un Lenguaje Específico de Dominio como "un lenguaje de programación o lenguaje de especificación ejecutable que ofrece, mediante notaciones y abstracciones apropiadas, un poder expresivo enfocado y usualmente restringido a un dominio particular del problema", el DSL introducido se presenta como un lenguaje de modelado compuesto de tres elementos: una *sintaxis abstracta*, que representa las abstracciones y relaciones del dominio; una *sintaxis concreta* que representa la forma en que los usuarios aprenderán y usarán el lenguaje; y una *semántica* que revela el significado de las expresiones [11].

La *sintaxis abstracta* del lenguaje de modelado es descrita a través de un metamodelo, que comprende un conjunto extensible de sensores típicamente usados en monitoreo estructural, y asocia objetivos de exploración de datos con un conjunto, también extensible, de alternativas de visualización gráfica y textual. La *sintaxis concreta* se define mediante un lenguaje visual, que representa gráficamente los conceptos del metamodelo y sus asociaciones. Finalmente, la *semántica* del DSL se define como un conjunto de reglas de transformación Modelo-a-Texto (M2T), que permiten generar automáticamente código

que implementa las alternativas de visualización modeladas con el lenguaje visual.

Del análisis con expertos en visualización de datos para monitoreo estructural basado en sensores, se identificaron dos grandes grupos de conceptos: (1) los *sensores*, que son la fuente de los datos que serán visualizados; y (2) los *elementos de visualización* que permiten representar dichos datos de forma gráfica, textual o tabular. El modelo de características presentado en la Fig.1 ilustra el resultado de este análisis.

Los sensores típicamente utilizados en monitoreo fueron clasificados en dos grupos: (a) *sensores básicos*, que entregan mediciones sobre una única variable, por ejemplo, acelerómetros (datos de vibración de la estructura), strain gauges (deformación), o inclinómetros (deformación respecto a la superficie), y (b) *sensores compuestos*, que entregan información sobre más de una variable y que pueden estar compuestos por sensores básicos, por ejemplo, estaciones meteorológicas, compuestas por termómetros, higrómetros, barómetros, etc.

Los elementos de visualización se agruparon en vistas (*views*), que representan espacios de interacción compuestos que dan soporte al análisis de datos mediante distintos elementos complementarios. Estos elementos pueden ser gráficos (*graphs*), tablas (*tables*) y tarjetas (*cards*). Los gráficos corresponden representaciones gráficas de datos desde distintas perspectivas y soportando diversos objetivos de análisis, como por ejemplo, gráficos de línea (*line*), de área (*area*), o de dispersión (*scatter*). Las tablas presentan datos en formato tabular, para facilitar el filtrado, comparación y ordenamiento de múltiples mediciones, mientras que las tarjetas destacan valores significativos del conjunto de datos analizado (máximos, promedios, medianas, etc.).

A. Sintaxis Abstracta del Lenguaje de Modelado

La sintaxis abstracta del lenguaje propuesto es descrita mediante un metamodelo, ilustrado en la Fig. 2. Sus metaclases representan los conceptos obtenidos a partir del análisis del dominio. Así, una visualización completa (metaclase *Vis4bridge*) está compuesta por una infraestructura de sensores (Fig. 2, izquierda) instalada en una estructura a monitorear. La metaclase *Sensor* se especializa en *BasicSensor* y *CompoundSensor* (sensor básico y sensor compuesto, respectivamente), las que, a su vez, se especializan en un conjunto extensible de sensores específicos. De forma complementaria, se consideran las metaclases *DAQ*, computador adquisidor de datos de sensores, y *UserGroup*, que representa un grupo de sensores definido por el usuario, de acuerdo a sus intereses de análisis.

Los conceptos relacionados con la visualización de los datos (Fig. 2, derecha), son agrupados por la metaclase *View*: las metaclases *Card* y *Table* representan las visualizaciones en tarjetas y tablas, mientras que *Graph* se especializa en distintos gráficos concretos, soportados en la actual versión del lenguaje y que, como submetaclases de Graph, pueden fácilmente complementarse con nuevos gráficos. Adicionalmente, las vistas consideran filtros (*Filter*) por fecha (*Date*) o de acuerdo al algoritmo Transformada Rápida de Fourier (*FFT*), frecuentemente utilizado en análisis de datos de sensores.

B. Sintaxis Concreta

La sintaxis concreta del lenguaje de modelado se expresa a través de un lenguaje visual, que permite la elaboración de alternativas de diseño para visualización de datos, con una terminología adecuada para un experto en el dominio, abstrayéndole de complejidades de implementación. En la confección del lenguaje visual, se buscó preservar la usabilidad en la definición gráfica de conceptos y en las interacciones necesarias para la confección de diagramas, considerando las heurísticas de usabilidad de Nielsen [12].

La representación gráfica de los sensores se asemeja a su apariencia real, mientras que los distintos tipos de gráficos son representados por iconos, con ejes y formas simplificadas. Tanto las vistas (*views*, agrupaciones de gráficos, tarjetas y tablas) como los grupos de sensores definidos por los usuarios (*userGroups*), se representan mediante conjuntos delimitados de las representaciones de sus respectivos componentes.

Para diseñar una visualización, se debe modelar gráficamente la asociación entre sensores y representaciones gráficas. La sintaxis abstracta (metamodelo) del DSL asocia un grupo de sensores (descrito por la metaclase UserGroup) con un gráfico (metaclase Graph) o una tarjeta (metaclase Card) incluidas en una vista (metaclase View). En tanto, la sintaxis concreta (lenguaje de modelado) permite expresar esta asociación mediante relaciones dirigidas desde grupos de sensores (*groups*) a los gráficos y tarjetas contenidos en una vista (*view*). Además, la presentación de gráficos varía cuando el grupo de sensores contiene uno o más sensores. La Fig.3 muestra un ejemplo del modelado gráfico de las visualizaciones: la Fig. 3a muestra un grupo con 1 sensor del tipo *strain gauge* asociado a un histograma, mientras que la Fig. 3b muestra un grupo de 1 sensor del tipo acelerómetro asociados a un gráfico de linea. Finalmente, la Fig. 3c muestra un grupo de tres sensores de tipo acelerómetro asociados a un gráfico de línea. Los respectivos iconos de los gráficos dan cuenta de la cantidad de sensores asociados, permitiendo al diseñador validar su modelo y anticipar su implementación.

C. Semántica

La semántica del lenguaje de modelado propuesto se expresa mediante un conjunto de reglas de transformación de modelo a código, que mapea las estructuras del metamodelo presentado a su implementación en lenguajes de programación específicos. Estas reglas fueron definidas utilizando el estándar MOF M2T [13]. Para la implementación de esta semántica, se escogieron dos lenguajes con bibliotecas de soporte a la visualización de datos, y se definieron sus correspondientes transformaciones de modelo a texto. La primera transformación se realizó con Python como lenguaje destino, utilizando la biblioteca de visualización *Plotly* y el framework *Dash* (https://plotly.com/dash/), para visualización en la web. La segunda transformación se definió para Javascript, pues permite generar visualizaciones web de forma nativa y con soporte para una visualización en páginas estáticas, mediante la biblioteca *ChartJS* (https://www.chartjs.org/).

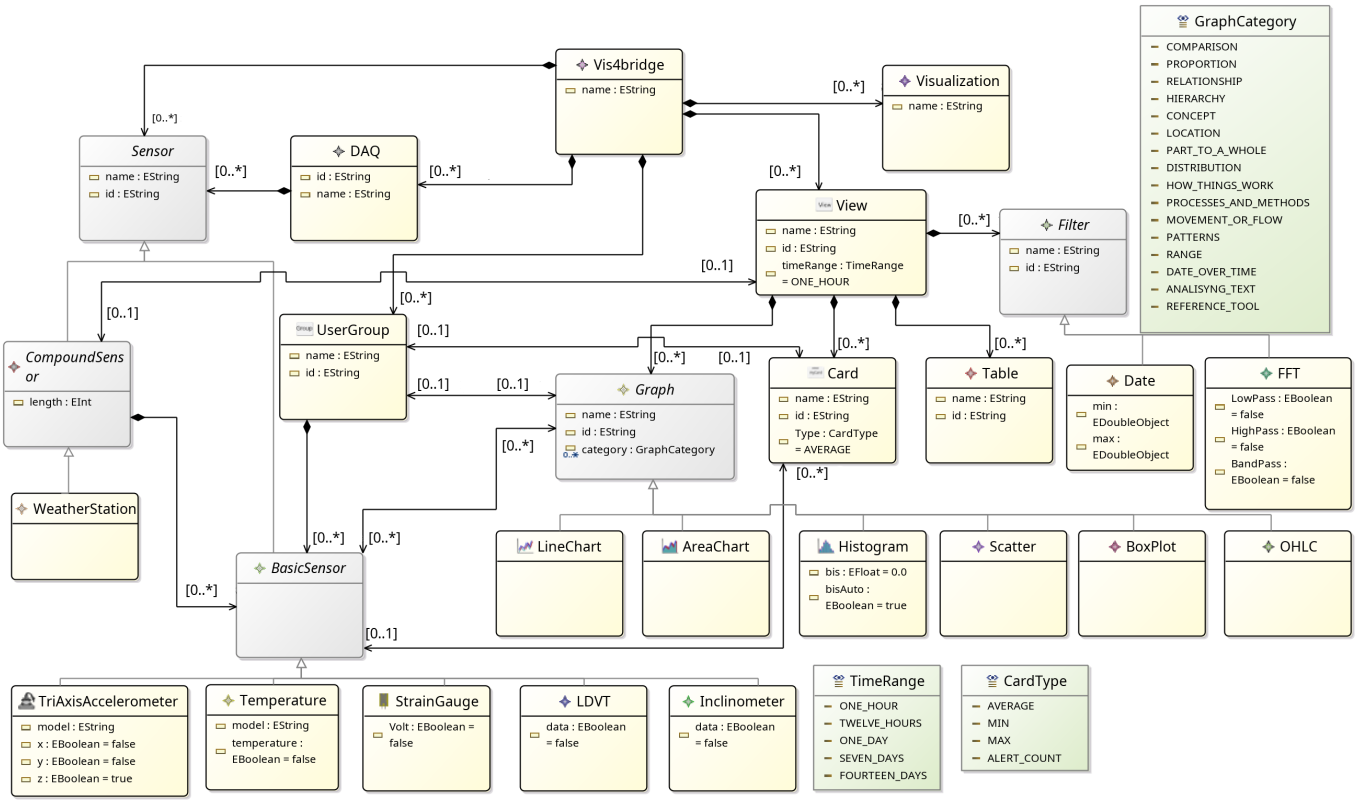


Fig. 2. Sintaxis abstracta (metamodelo de DSL) para visualización de datos de monitoreo estructural de puentes.

IV. Herramienta para la Generación de Visualizaciones

La implementación del DSL propuesto se realizó mediante el desarrollo de una herramienta de confección de diagramas basados en el lenguaje de modelado presentado. La herramienta, llamada *Vis4bridge* (acrónimo de "visualization for bridges"), permite la generación automática de código de la visualización modelada, en Python o Javascript.

La arquitectura del sistema presentado se describe mediante un Modelo C4 [14]. En este modelo se representa la arquitectura del sistema en 4 niveles que son:

1. Nivel 1 Contexto: Muestra una vista general del sistema que se centra en la interacción del sistema con los stakeholders y sistemas externos.
2. Nivel 2 Contenedores : Agrega detalle al nivel de contexto mostrando los contenedores (ejemplo de estos son las aplicaciones) que componen el sistema.
3. Nivel 3 Componentes: Muestra los componentes dentro de cada contenedor.
4. Nivel 4 Código: Muestra la representación del código de cada componente. Debido a la naturaleza de esta propuesta el nivel 4 no se presenta.

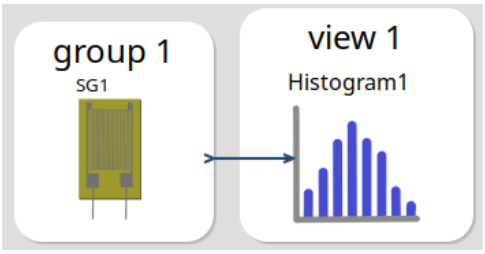
En cada Nivel se representa la interacción de los interesados del proyecto (stakeholders) en distintos niveles del sistema. Se identificaron como principales stakeholders a expertos en monitoreo estructural (Structural health monitoring expert), autoridades (Authority) y desarrolladores (Developer).

- Experto en monitoreo: Profesional dedicado al monitoreo estructural de puentes.
- Autoridad: Autoridades con la potestad de tomar decisiones referentes al uso y reparación de puentes.
- Desarrollador: Profesional que desarrolla software de visualización.

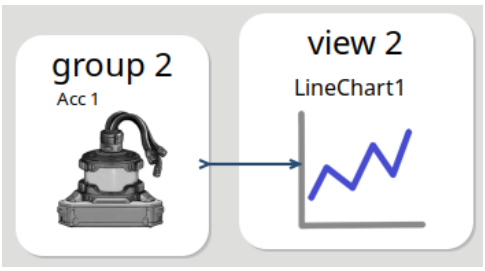
La interacción de los stakeholders con el sistema se muestra en la Fig. 4, correspondiente al nivel 1 del modelo C4.

La Fig. 5 muestra el nivel 2 del modelo C4, que describe la arquitectura de *Vis4bridge*. La herramienta se compone de dos contenedores principales: (a) el contenedor *Desktop Application* es el núcleo, que permite diseñar gráficamente las visualizaciones y generar automáticamente el código correspondiente. El contenedor *Data visualization Web App* implementa la arquitectura de la visualización resultante en el lenguaje destino, para su despliegue en la Web.

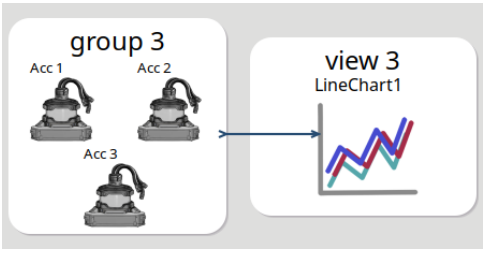
Las Fig. 6 y 7 muestran el tercer nivel C4 para la implementación en Python y Javascript, respectivamente. Como se aprecia, el cambio de lenguaje de destino solo afecta al contenedor *Data visualization Wep App* (a la izquierda de ambas figuras), pero no afecta la arquitectura del contenedor *Desktop Application* encargado de generar las aplicaciones, debiendo sólo adaptar el componente *Model to Text*. Entre los actores del sistema, el experto en monitoreo estructural y el desarrollador usan el contenedor *Desktop Application* para modelar su visualización de datos, mediante el componente *Graphical modeling tool*. Éste obtiene las reglas del dominio desde el componente *Meta-Model* y usa el componente *Java-Service* para acceder a funciones auxiliares. Una vez



(a) Strain Gauge asociado a un Histograma.



(b) Acelerómetro asociado a un Gráfico de línea.



(c) Múltiples Acelerómetros asociados a un Gráfico de línea.

Fig. 3. Sintaxis concreta (lenguaje gráfico) del DSL.

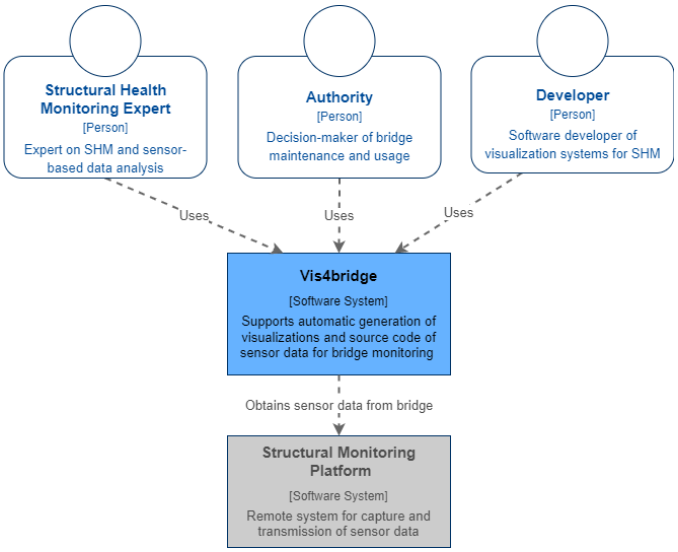


Fig. 4. Nivel 1 del Modelo C4 para Vis4bridge.

construido el diagrama de visualización, ambos actores generan su aplicación web de visualización de datos mediante el componente *Model to Text Transformation*. Este último componente obtiene el modelo correspondiente al diagrama de visualización confeccionado con *Graphical modeling tool* y lo transforma a una aplicación web funcional. En este

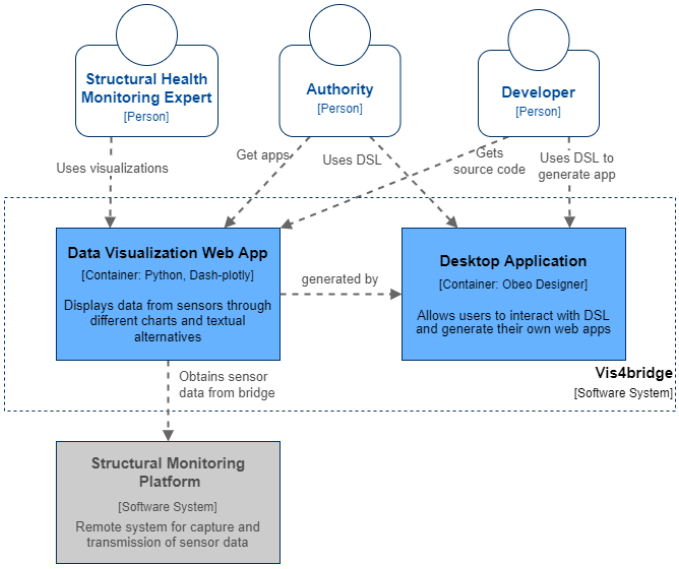


Fig. 5. Nivel 2 del Modelo C4 para Vis4bridge.

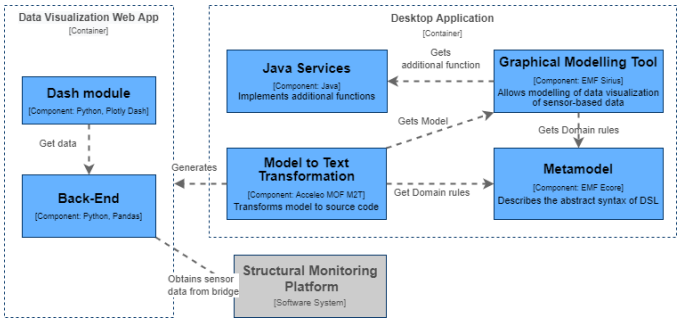


Fig. 6. Nivel 3 del Modelo C4 de Vis4bridge para Python.

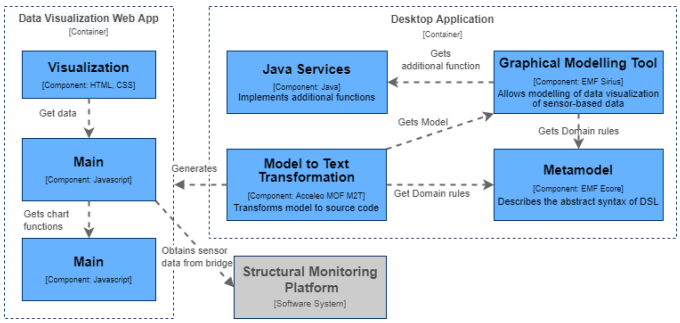


Fig. 7. Nivel 3 del Modelo C4 de Vis4bridge para Javascript.

punto, el experto en monitoreo estructural ya puede usar la aplicación generada, para la visualización y análisis de los datos y presentar sus conclusiones a la autoridad que gestiona el mantenimiento de los puentes o estructura relevante. Por su parte, el desarrollador puede adaptar el código generado a necesidades específicas de los stakeholders que no sean expresables por el lenguaje de modelado.

Vis4bridge fue implementada en el ambiente de desarrollo *Obeo Designer* (https://www.obeodesigner.com), basado Eclipse, que soporta la definición de metamodelos, lenguajes visuales y transformaciones de modelo a código. La herramienta, junto a su documentación e instrucciones de

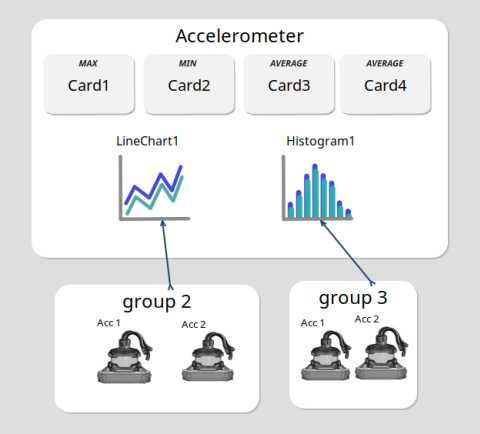
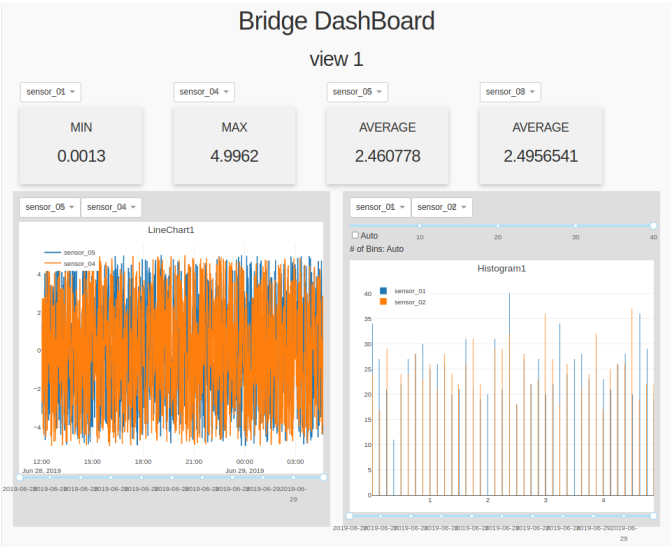
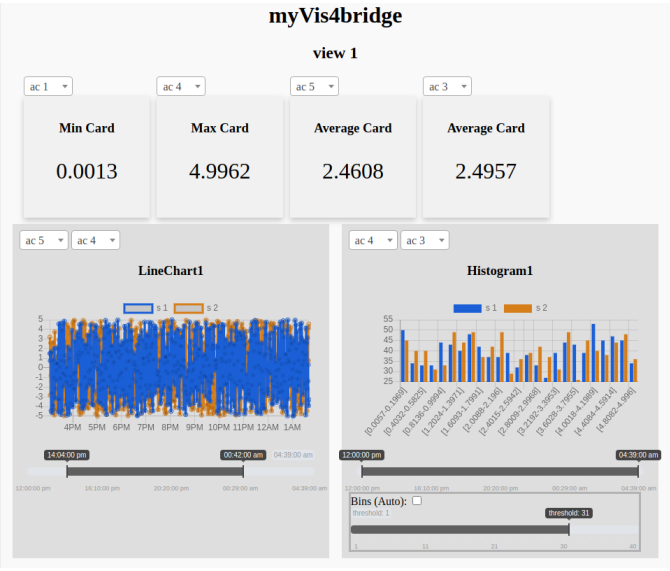


Fig. 8. Modelado gráfico de una visualización de datos en *Vis4bridge*.



(a) Resultado en Python



(b) Resultado en Javascript

Fig. 9. Visualizaciones generadas por *Vis4bridge* a partir de diagrama de la Fig. 8.

instalación y uso, se encuentra disponible para descarga en https://vis4bridge.gitlab.io/.

Como ejemplo de ejecución, la Fig. 8 muestra una captura del modelado de una visualización de datos de sensores en *Vis4bridge*. Este diagrama posee una vista con 4 tarjetas y dos gráficos, de línea e histograma, asociados a correspondientes grupos de dos acelerómetros cada uno. A partir de este mismo modelo, se generó su correspondiente implementación, tanto en Python (Fig. 9a) como en Javascript (Fig. 9b). Como se aprecia, ambas visualizaciones son desplegables en páginas web y son equivalentes. Sus diferencias se asocian al uso de las bibliotecas de visualización particulares de cada lenguaje, pero despliegan la misma información y ofrecen similares capacidades de interacción.

V. Evaluación

La propuesta dirigida por modelos presentada se orienta a facilitar que usuarios sin conocimientos de programación puedan desarrollar fácilmente visualizaciones de datos de sensores. Por ello, se evaluó la usabilidad de las distintas tareas del proceso de desarrollo, tanto en el modelado como en la generación de código, y el tiempo requerido para realizarlas.

A un conjunto de 17 ingenieros en las áreas de monitoreo estructural y desarrollo de software, se les presentó un conjunto de casos de estudio para entrenamiento en la herramienta y 3 casos para desarrollar en base a lo aprendido. Posteriormente, los participantes contestaron un cuestionario de usabilidad.

Debido a la crisis sanitaria del Covid-19, la capacitación y evaluación fueron realizadas en modalidad remota. Para ello, los participantes fueron referidos al sitio web de *Vis4bridge*, que cuenta con los siguientes elementos para dichos fines:

- Herramienta software para descargar. Disponible para sistema operativo Windows 7 o superior y Ubuntu 18.04 o superior (y derivados).
- Manual de instalación de la herramienta (en texto y video).
- Tutorial de uso de la herramienta (en texto y video).
- Escenarios para prueba de experiencia de uso.
- Cuestionario de usabilidad.

De esta forma, los participantes descargaron e instalaron la herramienta, asistidos por el manual de instalación, con prerequisitos de instalación (software y hardware), el software adicional necesario y cada uno de los pasos del proceso.

Antes de construir los escenarios, los participantes debieron realizar un tutorial de uso de la herramienta, mediante la exposición de un vídeo de 7 minutos y 48 segundos de duración. En el tutorial, se indica cómo crear un proyecto, cómo utilizar la paleta de herramientas y la ventana de propiedades. Adicionalmente, se dejó a disposición un tutorial en texto.

Posteriormente, los participantes realizaron las pruebas de experiencia de uso, que consisten en la realización de 3 escenarios de diseño de visualización de dificultad incremental, aumentando el número de elementos (gráficos, sensores o vistas) con los que el usuario debe interactuar. En cada escenario, se describió textualmente el modelo a diseñar en la herramienta y el resultado esperado.

Finalmente, cada participante contestó un cuestionario basado en encuestas de usabilidad propuestas por IBM [15], y

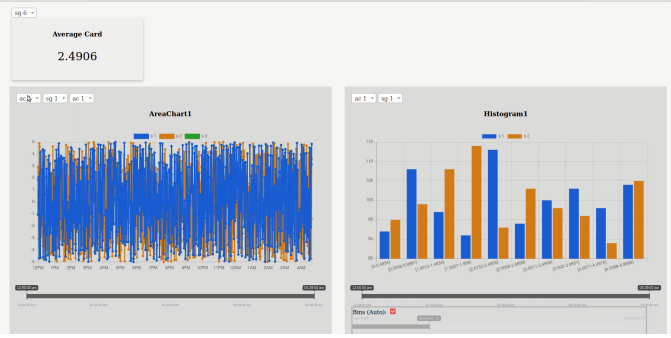


Fig. 10. Resultado esperado de uno de los escenarios de evaluación.

parcialmente adaptadas para este proyecto. El cuestionario está compuesto de 22 sentencias: 3 sobre satisfacción con respecto a las tareas realizadas y 19 sobre satisfacción con respecto al sistema. Cada sentencia consulta el grado de acuerdo a una sentencia de evaluación positiva, en una escala *Likert* de 7 niveles: *totalmente en desacuerdo*, *en desacuerdo*, *parcialmente en desacuerdo*, *ni de acuerdo ni en desacuerdo*, *parcialmente de acuerdo*, *de acuerdo* y *totalmente de acuerdo*. El estudio de IBM demuestra que una escala Likert de 7 niveles otorga mayor fiabilidad para encuestas de usabilidad que una tradicional de 5 niveles. El cuestionario aplicado se encuentra disponible en https://34pv.short.gy/encuestaUsabilidad.

A. *Escenarios*

Se plantearon 3 escenarios para la construcción de las visualizaciones. En cada escenario se plantearon las instrucciones en forma escrita y una imagen mostrando el resultado esperado. A continuación, se muestra el escenario de menor dificultad:

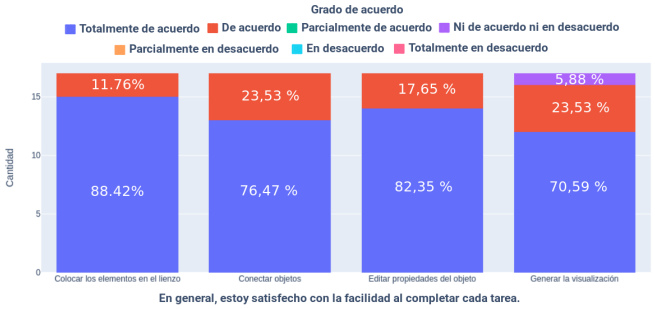
"Se solicita construir una visualización con las siguientes características:

- *Genere una vista (view) con un gráfico de área, un histograma y una tarjeta.*
- *Cree un grupo de sensores con 3 sensores a su elección. Conecte el grupo de sensores con el gráfico de área.*
- *Cree un grupo de sensores con un acelerómetro y un strain gauge. Conecte el grupo al histograma.*
- *Genere la visualización, con un resultado esperado como el que muestra la Fig. 10"*

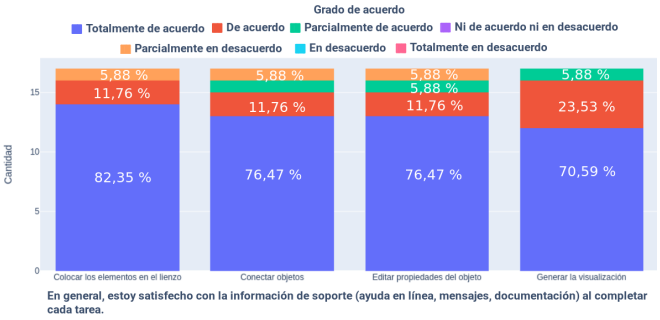
B. *Resultados*

Los resultados de la evaluación realizada se dividen en dos partes. La primera parte corresponde a las preguntas asociadas a la satisfacción del usuario al completar las tareas planteadas en los escenarios; la segunda parte, la satisfacción del usuario con respecto a la usabilidad del sistema *Vis4bridge*. Con respecto a las tareas realizadas en la creación de escenarios, la tarea mejor evaluada fue *Colocar los elementos en el lienzo*, en particular en el ítem *"En general, estoy satisfecho con la cantidad de tiempo que tomó completar cada tarea"* (ver Fig. 11a) donde el grado de acuerdo de todos los encuestados fue *Totalmente de acuerdo*.

Por otro lado, las tareas peor evaluadas fueron *Conectar objetos* y *Editar propiedades del objeto*. En el ítem *"En*



(a) Mejor evaluada



(b) Peor evaluada

Fig. 11. Resultados sobre satisfacción de usuario al completar tareas planteadas en los escenarios.

general, estoy satisfecho con la información de soporte (ayuda en línea, mensajes, documentación) al completar cada tarea" (ver Fig. 11b) un 76,5 % de los encuestados estuvo *Totalmente de acuerdo*, 11,8 % estuvo *De acuerdo*, 5,9 % estuvo *Parcialmente de acuerdo* y 5,9 % estuvo *Parcialmente en desacuerdo*.

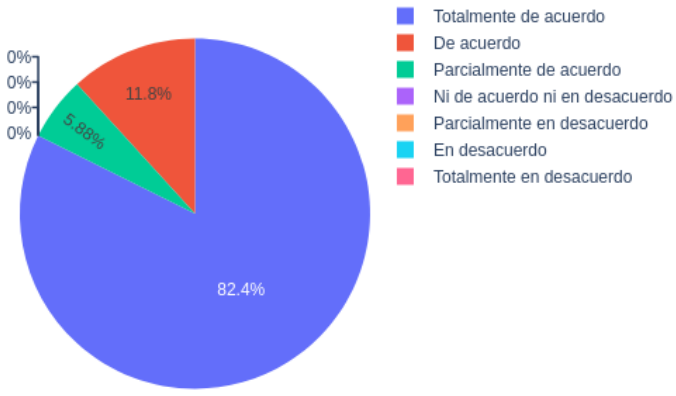
Con respecto al grado de satisfacción sobre la herramienta *Vis4bridge*, el ítem *"Fue sencillo utilizar este sistema"* fue el mejor evaluado, con un 82,4 % de los participantes manifestándose *Totalmente de acuerdo* con esta afirmación, 11,8 % *De acuerdo* y 5,9 % *Parcialmente de acuerdo*.

El ítem *"El sistema dio mensajes de error que me indicaron claramente cómo solucionar problemas"* fue el peor evaluado, con sólo un 29,4 % de los encuestados *Totalmente de acuerdo* con la afirmación y un considerable 11,8 % manifestándose en *Desacuerdo* o en *Totalmente en desacuerdo*. En las observaciones textuales, varios participantes reportaron no haberse enfrentado a errores que les permitiera pronunciarse adecuadamente sobre este ítem.

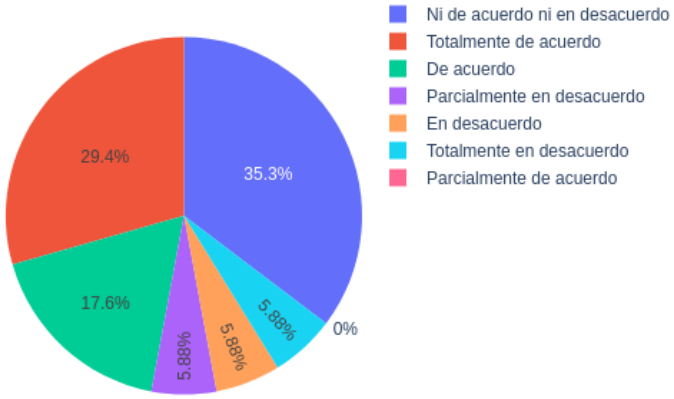
C. *Discusión*

La adopción de una aproximación dirigida por modelos para el desarrollo de visualizaciones de datos, en el contexto del monitoreo estructural basado en sensores, fue bien recibida por parte de los participantes de la evaluación. La dificultad de las tareas de modelado y generación de código fue bien evaluada, así como la usabilidad de la herramienta de apoyo.

Con respecto a la satisfacción del usuario al realizar la tarea *"Conectar objetos"*, el resultado es bueno, aunque 5,9 % de los encuestados manifestó estar *Parcialmente en desacuerdo* con la afirmación positiva con respecto a su usabilidad. Se detectó



(a) Mejor evaluada: Fue sencillo utilizar este sistema.



(b) Peor evaluada: El sistema dio mensajes de error que me indicaron claramente cómo solucionar problemas.

Fig. 12. Resultados sobre la usabilidad de tareas en herramienta de modelado.

confusión en los elementos destino de una asociación. Como soluciones posibles, se consideran la adición de mensajes que indiquen y expliquen la imposibilidad de conectar cierto tipo de elementos de forma directa. Una solución más permanente podría, previo análisis con los expertos en monitoreo y visualización, sugerir una actualización del metamodelo para dar soporte a acciones erróneas frecuentes, pero que se puedan permitir en futuras versiones. Por ejemplo, se podría proveer la asociación directa de un grupo de sensores a una vista completa, no sólo a un gráfico de ésta.

De modo similar, el resultado obtenido en la tarea *"Editar propiedades del objeto"* es favorable, pero un 5,9 % de los participantes se manifestó *Parcialmente en desacuerdo* con la sentencia positiva. En este sentido, se reportaron dificultades en el acceso a las propiedades de un elemento en un cuadro de interfaz separado. Aunque es un característica típica de interfaces de entornos de modelado, una probable solución es la posibilidad de acceder a dichas propiedades seleccionando el elemento gráfico correspondiente.

Con respecto a la insatisfacción expresada en el ítem *"El sistema dio mensajes de error que me indicaron claramente cómo solucionar problemas"*, es aconsejable realizar un test de usuario que permita registrar de forma no intrusiva la naturaleza y frecuencia de los errores cometidos, para mejorar la herramienta de modelado en términos de prevención y reporte de errores, y de recuperación a estado sin error.

A pesar de este favorable resultado, aún existen aspectos que merecen una discusión mayor con respecto al método de desarrollo propuesto y la herramienta desarrollada.

Uno de los temas que merece atención es la mantenibilidad de la implementación generada. ¿Qué tan fácil es modificar el código generado para ser usado por los desarrolladores? El código fuente generado para cada visualización es modular y se encuentra documentado, por lo que se estima que alcanza niveles de mantenibilidad adecuados para desarrolladores con experiencia en los lenguajes destino adoptados. Sin duda, esta mantenibilidad dependerá también del alcance de eventuales modificaciones y evoluciones del código. Por ejemplo, el cambio de la fuente de datos supone un menor esfuerzo que adaptar la visualización para combinar gráficos ya incluidos con otros nuevos. En cambio, una modificación de la configuración por defecto de la visualización elegida, que implique la inclusión de más parámetros en la invocación a la implementación correspondiente, o un complemento con texto explicativo de los gráficos generados, puede requerir un esfuerzo mayor. Se recomienda, en cualquier caso, un acceso a la documentación y un entrenamiento similar al propuesto en la evaluación, para los desarrolladores encargados de mantener el código.

Otra cuestión relacionada con el dominio específico del problema es la suficiencia de las opciones visuales consideradas para el ámbito de Monitoreo de Salud Estructural. Si bien la versión actual del metamodelo considera opciones visuales recogidas del análisis del estado del arte y de la técnica en este dominio, la flexibilidad del metamodelo y del enfoque de desarrollo propuesto permite añadir nuevas opciones gráficas que se adapten mejor a los requerimientos de stakeholders particulares, como también a diferentes contextos de análisis (un análisis de la señal en el tiempo puede requerir un gráfico de línea, mientras un análisis estadístico para periodos extensos puede sugerir el uso de histogramas).

VI. Conclusiones

El presente trabajo describe un marco de desarrollo dirigido por modelos para la visualización de datos, en el contexto de Sistemas de Monitoreo Estructural. Basado en la definición de un Lenguaje Específico de Dominio para la visualización de mediciones de sensores, se presentó un lenguaje de modelado a través del metamodelo que define su sintaxis abstracta, el lenguaje de modelado visual que describe su sintaxis concreta, mediante elementos visuales intuitivos para los expertos en monitoreo, y su semántica, con un conjunto de transformaciones de modelo a texto, que permiten asociar los elementos del metamodelo a código ejecutable. Todo este proceso está soportado por la herramienta Vis4bridge, que permite describir alternativas de visualización mediante un lenguaje gráfico, asociándoles diferentes tipos de sensores, y generando código ejecutable en, actualmente, dos lenguajes de programación (Python y Javascript).

Esta propuesta se evaluó exitosamente con datos provenientes de una infraestructura de sensores instalada en el puente La Mochita, Concepción, Chile, considerando un límite de tiempo de 2 semanas. Dicha infraestructura está compuesta por 18

acelerómetros triaxiales, con una frecuencia de muestreo de 120 Hz. En esta configuración, el volumen de datos generado por el sistema corresponde a 4,19 GB diarios. De esta forma, las gráficas generadas por el sistema, para un máximo de 14 días de muestreo, representaron a 58,66 GB en total.

Los experimentos realizados muestran que la aproximación de desarrollo propuesta recoge las ventajas del desarrollo dirigido por modelos de reducir los de tiempos de desarrollo. Sin embargo, no es incompatible con el desarrollo tradicional. Los lenguajes de destino seleccionados para las transformaciones de modelo a texto son comúnmente utilizados para la implementación de visualizaciones de datos, por lo que los resultados de la ejecución del proceso propuesto pueden ser mantenidos y modificados fácilmente.

En lo inmediato, se deberá trabajar en la mejora de aquellos aspectos de la herramienta reportados como insatisfactorios por los participantes de la evaluación. La dificultad en encontrar un destino válido para una asociación entre sensores y visualizaciones puede afectar negativamente la adopción de la herramienta, u ocasionar errores en el modelado que se propaguen a la implementación resultante. Además, se debe mejorar la usabilidad de la edición de propiedades textuales de los elementos gráficos seleccionados, para asegurar que todos los datos necesarios para la transformación de modelo a código hayan sido ingresados de forma correcta. Finalmente, la gestión de errores de uso amerita la realización de una nueva evaluación de usuarios, para planificar e implementar las reparaciones necesarias a la herramienta.

Como trabajo futuro, se plantea la extensión del conjunto de visualizaciones de datos y de sensores considerado por el DSL, de acuerdo a los requerimientos de análisis de expertos en Monitoreo de Salud Estructural. El carácter modular del metamodelo permite, además, adaptar esta propuesta para la generación de visualizaciones de datos en otros dominios.

Otra línea de trabajo que complementa esta propuesta consiste en la evaluación de distintas alternativas arquitectónicas de las implementaciones obtenidas, que permitan reducir los tiempos de respuesta de las visualizaciones obtenidas. El despliegue de un volumen de datos del orden de decenas de gigabytes desafía las capacidades de computación disponibles para usuarios comunes. Por ello, si bien la solución propuesta no está condicionada por un límite máximo en la cantidad de datos a desplegar, la eficiencia de la implementación resultante puede variar de acuerdo a la opción arquitectónica adoptada. Actualmente, estamos trabajando en la evaluación de alternativas arquitectónicas para un despliegue más eficiente de una misma cantidad de datos y su incorporación en el proceso de generación automática de código.

AGRADECIMENTOS

Este trabajo fue financiado por la Agencia Nacional de Investigación y Desarrollo de Chile, Gobierno de Chile, a través de proyecto FONDEF, IT18I0112.

REFERENCIAS

[1] C. R. Farrar and K. Worden, "An introduction to structural health monitoring," *CISM International Centre for Mechanical Sciences, Courses and Lectures*, vol. 520, no. 1851, pp. 1–17, 2010.

[2] J. Morales Valdez, L. Alvarez-Icaza, and J. A. Escobar, "Online identification system for damage location in building structures," *IEEE Latin America Transactions*, vol. 17, no. 08, pp. 1283–1290, 2019.

[3] C. Ledur, D. Griebler, I. Manssour, and L. G. Fernandes, "Towards a domain-specific language for geospatial data visualization maps with big data sets," in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–8, 2015.

[4] S. F. Masri, L.-H. Sheng, J. P. Caffrey, R. L. Nigbor, M. Wahbeh, and A. M. Abdel-Ghaffar, "Application of a Web-enabled real-time structural health monitoring system for civil infrastructure systems," *Smart Materials and Structures*, vol. 13, pp. 1269–1283, dec 2004.

[5] S. L. Desjardins, N. A. Londoño, D. T. Lau, and H. Khoo, "Real-Time Data Processing, Analysis and Visualization for Structural Monitoring of the Confederation Bridge," *Advances in Structural Engineering*, vol. 9, no. 1, pp. 141–157, 2006.

[6] I. Khemapech, W. Sansrimahachai, and M. Toahchoodee, "A real-time Health Monitoring and warning system for bridge structures," in *TENCON 2016 Conference*, pp. 3010–3013, IEEE, jun 2016.

[7] M. A. Fabrício, F. H. Behrens, and D. Bianchini, "Monitoring of industrial electrical equipment using iot," *IEEE Latin America Transactions*, vol. 18, no. 08, pp. 1425–1432, 2020.

[8] K. Smeltzer and M. Erwig, "A domain-specific language for exploratory data visualization," in *Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, GPCE 2018, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2018.

[9] M. Brambilla and E. Umuhoza, "Model-driven development of user interfaces for IoT systems via domain-specific components and patterns," in *ICEIS 2017 - Proceedings of the 19th Int. Conf. on Enterprise Information Systems*, vol. 2, pp. 246–253, SpringerOpen, dec 2017.

[10] A. Van Deursen, P. Klint, and J. Visser, "Domain-specific languages: An annotated bibliography," *ACM Sigplan Notices*, vol. 35, no. 6, pp. 26–36, 2000.

[11] A. Rodrigues Da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Computer Languages, Systems and Structures*, vol. 43, pp. 139–155, 2015.

[12] J. Nielsen, "10 usability heuristics for user interface design," *Nielsen Norman Group*, vol. 1, no. 1, 1995.

[13] OMG, "MOF Model to Text Transformation Language, v1.0," Jan. 2008.

[14] S. Brown, "The c4 model for visualising software architecture," *https://c4model.com/*, 2018.

[15] J. R. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human-Computer Interaction*, vol. 7, no. 1, pp. 57–78, 1995.

Braulio Quiero es Ingeniero Civil Informático de la Universidad Católica de la Santísima Concepción, Chile (2017) y Magíster en Ciencias de la Computación de la Universidad de Concepción, Chile (2021). Su trabajo se orienta a la generación automática de código, interfaces humano-computador, y visualización de datos.

Gonzalo Rojas es Profesor Asistente del Departamento de Ingeniería Informática y Ciencias de la Computación de la Universidad de Concepción, Chile, y Doctor en Ingeniería de Software por la Universidad Politécnica de Valencia, España (2008). Su trabajo se orienta a arquitectura de software, sistemas de Big Data, visualización de datos y calidad de software.