BISEL: A Specialised language for data integration on infrastructure information models

Beatriz Patusco Neto
Instituto Superior Técnico, Universidade de Lisboa, Portugal
beatriz.p.neto.08@gmail.com

ABSTRACT

The activity of combining sensor data with space, other physical and functional information extracted from digital Building Information Modeling (BIM) is becoming a common practice in buildings performance analysis and monitoring. However, this integration is complex. Besides the inherent complexity of data integration itself, there is an additional challenge which concerns how to enable non-programmers domain experts to exploring this integration effectively. Although some approaches integrating BIM and real-time sensor data already have been proposed, none of them enables domain experts to effectively exploit the data because of a lack of appropriate abstractions for domain experts. Also, infrastructures along their lifecycle need refinements in their monitoring systems and these solutions would require an extra and heavy development effort to support them.

This thesis proposes a high-level Domain Specific Query Language capable of rising both the level of abstraction and adaptability, which will enable domain experts to formalize real-time queries regarding the integration of BIM with sensor data. We use a model-driven top-down approach where the proper abstractions and requirements are captured involving domain experts through Focus Group meetings. By using model-driven technologies with code-generation facilities, the solution is automatically generated and refined with almost zero development effort. Our solution is validated according to an evaluation methodology focused on usability and flexibility attributes using a real-world BIM models of a sensitised water dam infrastructure.

1. INTRODUCTION

Since sensors have become increasingly easy to install at reduced prices, the number of units installed has been increasing steadily [45]. The Internet of Things (IoT) is the visible face of this trend. Consequently, large facilities and complex infrastructures that require continuous monitoring are increasingly supported by management tools that rely on these sensor data to optimise their management processes. However, when analysing the performance of complex infrastructures, sensor data alone is not enough. Infrastructures' structural properties and behaviour must be analysed to predict functionality and collapse scenarios associated with infrastructure construction and exploitation. Therefore, sensor data has to be combined with built environment information that can be extracted from information models such as BIM [45].

Several research challenges concerning the integration of BIM and sensor data have been identified [2, 13, 38]. First, processing real-time sensor data involves technical knowledge regarding sensor data fusion and windowing. Second, it requires expert knowledge in Civil Engineering and, of itself, modelling standards are not often followed correctly. Finally, the above complex domains have to be combined.

Many approaches have been proposed to overcome the referred challenges [3, 14, 15, 17, 21, 37], including plugins, engines, and software solutions. Yet, most approaches lack practical validation [2] or were developed for a specific context such as Fire Safety Control [15], Building Automation [14, 17], or Energy Management [10] and, therefore cannot be easily generalised. Recently, [2] developed a Domain-Specific Language (DSL) that integrates real-time queries over sensor data while embedding BIM model information [2]. While the approach has demonstrated significant gains in expressiveness and ease of use, this DSL is still oriented to software developers and does not empower domain experts with tools that enable them to exploit the data effectively [2]. Indeed, when end-users are non-developer experts, a solution with higher abstractions by using familiar notations and concepts is required to handle the specification of queries involving sensor data. However, find the appropriate language abstractions are by itself a challenge, and we must be aware that one iteration might not be enough because after using the tool domain experts may require refinements.

We hypothesise that creating a high-level DSL with the appropriate domain abstractions, it is possible to allow domain-experts to explore the integration between sensors and BIM models. BIm Sensor Exploration Language (BISEL) will be developed using a top-down model-driven approach following development methodology in literature. Another innovation of this research work is that the DSL' proper abstractions and requirements are captured involving domain experts through a Focus Group technique. Additionally, by using model-driven technologies with runtime support for automatic code generation facilities, we expect to reduce the development effort to almost zero and provide to end-user a solution flexible to refine and improve.

To demonstrate that the proposed solution is accessible for non-programmers domain experts and suitable to a diversity of application domains, a Case Study consisting of the implementation of several queries requiring the integration between BIM and sensor data was performed. BISEL was validated according to a DSL usability evaluation recommended by the scientific community to assess the effectiveness, efficiency, and satisfaction of the users when using BISEL. By including Software Engineers in the assessment, it was also possible to compare the performance during query specifi-

cation of BISEL against an existing alternative solution.

2. BACKGROUND

2.1 Domain Specific Languages

Domain-specific languages are specialised languages that offer constructs and vocabulary to model a problem explicitly or, in some instances a solution for a problem [22]. DSLs can ease the design and implementation of a system, by reducing the gap between the problem domain and the solution domain [6, 9]. The claim is that the closer we get to fill this gap, the closer we are to increase the user's productivity [6].

According to [42] one of the value propositions of DSLs is that they provide to non-programmers a "*clean, custom, productive environment that allows them to work with languages that are closely aligned with the domain in which they work*". DSLs have been successfully used in many application domains including the constructions field, health monitoring applications, automotive, cooling algorithms in refrigerators, and typesetting [34, 35, 42].

2.2 Model-Driven technologies

The abstract and concrete syntax of DSLs can also be modelled. This is called meta-modelling [18, 20]. Meta-modelling provides the ability to capture explicitly key features of a language in a platform-independent way. It enables language definitions to become more straightforward since domain-specific concepts and abstractions can be represented directly within the language [18, 20]. However, the effort that goes from the definition of the meta-model into producing a language definition and the corresponding editor can become time-consuming on development without automated tools [18]. Model-Driven Development (MDD) technologies such as Eclipse Modeling Framework (EMF) or Graphical Modeling Framework (GMF) provides runtime support from which you can automatically generate all the executable code from models [39]. These frameworks reduce the time spent on development, allowing language developers to focus on design decisions relevant to the domain rather than in the implementation.

EMF is a model management framework that unifies three important technologies, Java, XML and UML, by allowing the specification of meta-models in any of these formats. EMF code generators produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor [27, 39]. The GMF is a framework used to implement a graphical editor for EMF-based modelling languages [30].

2.3 Sensor-based Monitoring Software

In applications that deal with sensors, data is modelled best as transient data streams instead of persistent relations [12, 23, 25]. Applications that monitor continuous data streams are used to support a timely decision making process. This kind of applications can be found in many fields such as financial monitoring of stock, tracking applications used to monitor the locations of items of interest, or for security in physical facilities [11].

Monitoring applications can possibly monitor multiple data streams (possibly in the order of thousands), leading to processing rates bigger than ones supported by a traditional Database Management System (DBMS). This aspect calls for the need of using Data Stream Management Systems (DSMS) [5] as an alternative to provide timely results. [5, 12, 23]. A DSMS processes input data streams from a wide diversity of sources and produce a new output stream as a result.DSMS also offers a flexible query processing enabling the processing operations to be specified using queries.

Sensor-based monitoring applications have the following characteristics.

1. They use approximate summary structures to deal with the impossibility of storing a complete data stream in memory.
2. The data required is not merely the last recently reported values but also historical data.
3. Real-time monitoring applications must process data in a timely way, reacting quickly to unusual data values. This way, it is possible to detect anomalies in due time enabling users to react.

3. RELATED WORK

3.1 Real-time data applications to BIM

The emergence of BIM and, as a consequence, approaches that combine BIM and real-time data is promoting the rise of the concept of BIM-based real-time monitoring and management. To understand the current state-of-art of the integration between BIM and real-time data, a literature review on the most significant recent approaches regarding the problem was performed.

After analysing the existing solutions, we can conclude that the use of BIM models on application domains related to smart buildings has not been fully explored. Most of the proposed solutions regarding this integration lack practical validation or were created for a specific application context, and thus not easily generalisable. Therefore, the development of these solutions is associated with a high level of complexity, lacking a standard solution adaptable to several application domains to retrieve real-time information related to BIM models. This conclusion is supported by the updated literature review presented in Table 1. All approaches were classified according to how they were implemented, and we also analysed if the solution remains in the theoretical domain, not having yet been implemented, as well as if the approach was intended to be a framework or a plug-in for an external tool. Besides, all the proposed solutions are highly dependent on a particular domain like Building Automation (BA), Energy Management (EM), and Fire Control (FC) which reinforces the need of a generalisable standard solution that simplifies the creation of applications.

3.2 Querying Approaches

To deal with a large amount of data regarding both BIM and sensor data, several approaches on BIM querying have been advanced by the scientific community. We searched for existing DSLs in Civil Engineering field regarding BIM models or/and real-time sensor querying, namely: BERA [32], BIMQL [34], GPL4SRE [1], and Building Information Modeling Sensor Language (BIMSL) [2] are elicited. As presented in Table 2, we focused on two main characteristics. First, we examine if the solution deals with real-time data and BIM models. Next, we determine if the language requires software and BIM formats skills to be used by end users.

Reference	Implemented	F/P	Validated
Building Automation			
Chen et al. [14]	•	**F**	–
Ciribini et al. [17]	–	**F**	–
Howell et al. [26]	•	**F**	•
Energy Management			
Niu et al. [36]	•	**P**	–
Choi et al. [16]	•	**F**	•
Brundu et al. [10]	•	**F**	•
Fire Control			
Li et al. [33]	•	**P**	•
Cheng et al. [15]	•	**F**	•
Health and Safety			
Park et al. [37]	•	**F**	•
Riaz et al. [40]	–	**F**	–

Table 1: Comparison of current Real-Time Data applications to BIM referred in the literature. Each approach is classified according to whether it was implemented, was developed as a framework, was designed as a plug-in for another tool, and whether it has been adequately validated. ● Related, – Not Related, **F** Framework, **P** Plug-in

	Domain		User Skills	
DSLs	Real-time sensor data	Information models	Query Languages	BIM
BERA [32]	–	●	○	○
BIMQL [34]	–	●	●	●
GPL4SRE [1]	●	–	●	–
BIMSL [2]	●	●	●	●

Table 2: List of DSLs classified according to its relationship with BIM and real-time data processing, as well as with the required end-user skills. ●Yes. ○No. – Unrelated.

From the analysis, only BERA provides an intuitive notation, easy to learn and apply by non-programming users. For example, in the query specification, we can use the keyword "Space" instead of "IfcSpace" in Industry Foundation Classes (IFC). However, BERA does not process real-time data. On the other hand, only BIMSL handles the high-complexity required to integrate BIM and real-time sensor data. Even so, BIMSL elements have similar semantics to Structured Query Language (SQL) elements, what makes the formulation complex for non-programmers because of the lack of software skills.

So, we can conclude that no standard solution empowers Domain Experts to specify real-time queries regarding buildings and infrastructures showing the pertinence of our work.

4. METHODOLOGY

The language development follows two phases, summarised in Figure 1. First, a Focus Group research is conducted to find the proper abstraction and elicit functional and technical requirements. Then, the second phase of our work, named DSL development life Cycle, relates to the DSL development and follows an iterative methodology adapted from Barišić et al. [9] approach.

The Focus group is divided into three main stages, namely (1.1) Study preparation, where the group of experts is selected and meetings are formulated, (1.2) Meetings, until we reach the proper abstract concepts and requirements, and (1.3) Study Results, were results are finally achieved and used to formulate a case study together with experts. The Focus group is described in detail in Section 5.

The DSL development life Cycle is composed of four main stages, namely (2.1) Domain analysis, (2.2) Language design, (2.3) Language implementation, and (2.4) Language evaluation. The DSL will be developed atop of standardised meta-modelling infrastructures, EMF, which provides reflective mechanisms to encode its abstract syntax, and GMF to its graphical syntax and creating Java code from it.

The language development approach can be summarised in the following steps:

1. Considering the focus group results, we start by performing a domain analysis, where domain abstract concepts, domain knowledge and functional and technical requirements are identified. These findings will support the language design.
2. In the next stage, we design our DSL by describing its abstract and concrete syntax using MDD techniques. These definition includes element' properties, rules, relation and the expected behaviour of language elements. The language semantics its also defined.
3. In the DSL implementation and deployment stage, we will define and develop a detailed target platform of the solution and its subsequent implementation. In this stage transformations of DSL code, written in a concrete syntax, to programming language code are performed using atomated frameworks.
4. The final step is the evaluation stage and refers to validation of the DSL. The evaluation will be carried out by domain expert users in the context of a case study to be developed by us for this purpose according to an evaluation methodology focused on usability and

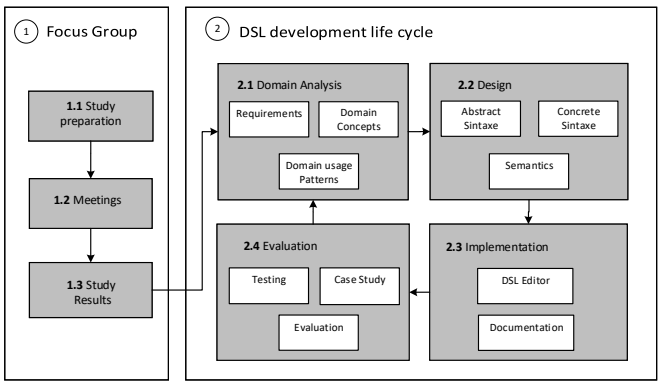

Figure 1: The Methodology stages. First, a Focus Group study will be conducted to extract functional and technical requirements of the DSL and then the DSL is developed following an iterative methodology adapted from Barišić et al. [9]

flexibility attributes.

5. FOCUS GROUP

The Focus Group is a qualitative research technique frequently used to gain in-depth understanding about a given issue or topic of interest through an informal group discussion [28, 44]. Using this technique, researchers learn from selected individuals while they debate a set of questions with each other and comment on each other's points of views [24, 41].

In our case, we use this technique to explore the domain under study meticulously. Indeed, in addition to discover the fundamental monitoring operations and the proper abstractions to define the language. We also intend to validate the complexity of integrating BIM and real-time data and understand the shortcomings of current monitoring processes. For this reason, was vital for the discussion the selection of civil engineers that are acquainted with sensor data analysis and with at least a little knowledge in BIM.

The Focus Group was organised in the following parts:

1. Validation of Research Context
   - Analyse sensor device complexity of the infrastructure and discuss how spatial data may benefit monitoring processes.
2. Elicit main processes and KPIs
   - Elicit the current practices concerning sensor data processing in infrastructures and identify the best practices for presenting the sensor data analysis results in the infastructures's monitoring systems
3. BIM for sensor data processing
   - Discuss among the group the benefits of integrating BIM in the monitoring systems and identify possible processes that can be optimised with the integration of BIM

To help us conduct the meetings we previously define a guide were we formalise questions for each topic of the focus group.

At the end of the sessions, based on the elicited requirements, possible queries combining BIM and sensor data in infrastructures will be formalised. The goal is, By planning a case study composed of these queries, we pretend that, at the end of implementation, the domain experts can evaluate the solution usability and ask for improvements until reaching the ideal monitoring platform.

We present the focus group results in the DSL Development section, more specifically, in Domain Analysis (Section 6.1).

6. DSL DEVELOPMENT

6.1 Domain Analysis

In the first stage of DSL development the domain concepts and knowledge, including domain abstractions and existing constraints that will support the language design, will be elicited. The domain analysis was conducted with domain experts collaboration thought focus group meetings in order to find the proper language abstractions, identify concepts, terms, and expressions relevant to the problem solution. As a result, the output of Domain Analysis stage will be a Domain Model, which represents common and varying properties of the system within the domain.

Along the focus group meeting, the group identify the observation system's main processes of analysis. These processes are identified from the technical-scientific knowledge available on dam's main components, namely: *(i)* Actions, *(ii)* Structural properties, and *(iii)* Responses. Indeed, from the actions associated with external or environmental factors and his structural properties, it is possible to predict the dam's structural and thermal reactions. Besides, from the interpretation and structural behaviour of similar dams, it is possible to predict functionality and collapse scenarios associated with his construction and exploitation. These scenarios are used to identify the observation system's main processes of analysis. Analysing this scenarios it was also possible to identify the quantities that our platform must analyse and the methods of observation chosen for each one. The quantities are : Displacement, Reservoir water level, Temperature, Stress/strain, Infiltration/inflow and Uplift pressure.

At the end of this stage, it was possible to conclude with the group that solution must support queries that comprise the following requirements:

1. Analyse the evolution of the different quantities, which are bounded by a window
2. Obtain measures snapshots
3. Correlate measurements from different quantities
4. Locate sensors by proximity to other IFC objects or properties
5. Apply aggregation functions on a result-set
6. Control or stabilise the rate at which events are output, and suppress output events
7. Apply filters to records and extract only the records that fulfil a specified criterion

6.2 Language Design

The Domain Analysis stage is followed by Language Design. In this stage we start by define the language abstract and concrete syntax which includes the definition of language elements, rules, and relations. Then, we describe the expected behaviour of language elements and the stage finishes with the definition of the language semantics.

6.2.1 Abstract Syntax

The Design stage started by applying the results of Domain Analysis in order to describe the language elements that compose the language, independently of their future presentation or meaning. The language abstract syntax is defined to fully understand the language elements, the relations that exists between them and well-formedness rules that indicates how elements should be appropriately combined.

The BISEL abstract syntaxt was described using a metamodelling language from EMF, Ecore. Ecore lets us define the meta-model using object-oriented constructs such as classes, associations and generalisations. The elements of the presented meta-model are described in Table 3

Regarding the language grammar analysis, a relevant set of rules and constraints was defined using Epsilon Validation Language (EVL). EVL is a model validation language which use Epsilon Object Language (EOL), an imperative model-oriented language for creating, querying and modifying EMF models, as an expression language. EVL supports

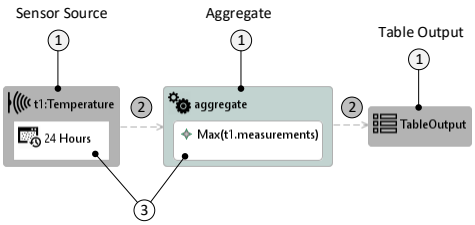

Figure 2: Graphical representation of some of the BISEL elements. The Sensor Source, Aggregate, and Table Output elements are represented as nodes (1), the relationships between elements as edges (2), and some of the elements properties as compartment nodes (3)

both intra and inter-model consistency checking, constraint dependency management and specifying fixes that users can invoke to repair identified inconsistencies. EVL is integrated with EMF and as such, EVL constraints can be evaluated from within the generated language editors and generate error markers for failed constraints [29].

6.2.2 Concrete Syntax

After describing BISEL abstract syntax, the next step is to define the BISEL concrete syntax. BISEL language elements will be represented using a graphical notation. Indeed, graphical languages are more effective in conveying information to non-IT experts and the learning curve is steeper for graphical languages [19]. The Concrete syntax establishes the concrete visual representation of language elements, defining that a certain entity should be represented by a specific geometric shape, defining the layout and spatial relationships.

The BISEL concrete syntax was defined using GMF, a framework within the Eclipse platform that provides a generative component and runtime infrastructure for developing graphical editors based on the EMF. BISEL queries will be represented by graphs consisting of nodes and directed edges that connect the nodes. Each BISEL element is represented as a node and uses special icons and tailored symbols to render elements depending on their type.

The association between the graphical representation and the abstract syntax was defined using *EuGENia*, an incubation project from Eclipse Epsilon used to reduce the complexity of developing an editor in GMF, by simply using annotations in the language meta-model. This process will be described in detain in the BISEL Editor implementation (Section 6.2.5).

6.2.3 Language Semantics

The next step after defining the abstract and concrete syntax is the definition of the formal semantics. In our case, we will define our language by means of algebraic operators that are very well understood and deeply studied in the field of streaming data.

- **Algebra Operators**:

The time domain $\Omega_{\leq}$ is a discrete ordered set of timestamps, also called instants, that captures application time (*viz.* System time).

A stream S is a potentially infinite set of elements, e_t^S, where e is an event or observation of type T, and $t \in \Omega_{\leq}$ is the associated timestamp. For a given stream S_T, the attributes of the events of S_T are represented by $A(T)$ and their corresponding domains by $D(T)$. Thus, an event e_t^s is an assignment function $e_t^s : A(T) \to D(T)$ that maps each distinct attribute of S_T to the corresponding value at instant t. Wherever S is understood, e_t^s will be simply written as e_t. The set of all streams of type T is represented by $\mathbb{S}_T$.

In this section first introduce our basic set of operators consisting of: filter (σ), map (μ), union ($\cup$), cartesian product ($\times$), and then our composite operators, built of the basic ones, such as join ($\bowtie$), and window (ω).

Projection Let $L \subseteq T$ be a subset of attributes. The projection $\pi_L : \mathbb{S}_T \to \mathbb{S}_L$ produces a stream where events have less attributes. For a given stream $S \in \mathbb{S}_T$, the projection operator is defined as:

$$\pi_L(S)\{e_t^s[L] \mid e_t \in S\} \tag{1}$$

where:

$$e_t^s[L](a) = \begin{cases} e_t(a) & \text{if } a \in L \\ \perp & \text{otherwise} \end{cases}$$

Filter Let C be a predicate over elements of type T. A filter $\sigma_C : \mathbb{S}_T \to \mathbb{S}_T$ produces a stream with all events that satisfy the condition C. Formally:

$$\sigma_c(e_t)\{e_t \mid C(e_t) \text{ is true}\} \tag{2}$$

Map Let $f : D(T) \to D(U)$ be a function that converts values of type T into values of type U. The map operator $\mu_f : \mathbb{S}_T \to \mathbb{S}_U$ defines a stream from another by applying a given mapping function to S_T such that $e_t^U = f(e_t^S)$. The mapping operator is defined as:

$$\mu_f(S)\{e'_t \mid e'_t = f(e_t) \wedge e_t \in S\} \tag{3}$$

Union The union $\cup : \mathbb{S}_T \times \mathbb{S}_T \to \mathbb{S}_T$ defines a stream with all elements of two streams of compatible types. The multiplicity of a tuple at instant t in the output stream results from the sum of the corresponding multiplicities in both input streams. The union operator is defined as:

$$S_1 \cup S_2\{e_t \mid e_t \in S_1 \vee e_t \in S_2\} \tag{4}$$

Cartesian Product The Cartesian product $\times : \mathbb{S}_T \times \mathbb{S}_U \to \mathbb{S}_{TU}$ of two streams combines the elements of both whose tuples are valid at the same time instant.

$$S_1 \times S_2\{e_{t_1} \cdot e_{t_2} \mid e_{t_1} \in S_1 \wedge e_{t_2} \in S_2\} \tag{5}$$

where $S \cap U \neq \emptyset$

Aggregate The aggregate $g : 2^{D(T)} \to D(U)$ function performs a calculation on sets of elements of type T, and returns a single value of type U.

$$\begin{aligned} {}_L\alpha_g(s)\Big\{ e' \mid e'_t[L] \in \pi_L(S) \wedge \\ e'_t[\cup] = g\Big(\{e \in S \mid e_t[L] = e'_t[L]\} \Big) \Big\} \end{aligned} \tag{6}$$

Element	Description
BIM source	Used to select the working BIM source. If this element is not used in the query, the BIM project is selected based on the contained information in the configuration file *BIMProperties*
Sensor Source	Used to selects the quantity that is intended to analyse along the query
Retention Period	Due to the real-time nature of sensor data, the user can obtain historical data by creating sliding windows using this element. The window selects a sub-finite part of the stream, instead of the entire stream based on time or a certain number of events. This element can be used, for instance, to process all tuples in the next 20 minutes. This window can be batched.
Filter	Allows the specification of any condition used to extract only the records that fulfil a specified criterion. If the condition is satisfied, then just a specific record from the sensor source is returned.
Location Filter	Special filter to query BIM models. Used to locate sensors and other spatial objects in the BIM model, using conditions.
Merge	Used to correlate measures from two Sensor Sources by specifying a merge condition.
Output Control	Used to control or stabilise the rate at which events are output and to suppress output events.
Aggregate/Computation	Used to specify how to aggregate data provided by the Source. Contains functions that allow performing calculations on sets of data, returning the result value of executing the given task. An example of a possible utilization is to specify that we want the hourly average of the data provided by the Sensor Source element.
Sort	Sort a result-set by one or more of its columns, in ascending or descending order.
Table/Graph Output	This elements are used to specify the form of presenting the query results

Table 3: Summary of language elements. These elements have been defined by means of the concepts, models, and requirements identified during Domain Analysis (see Section 6.1).

Join (Merge) The join $\bowtie$: $\mathbb{S}_T \times \mathbb{S}_U \rightarrow \mathbb{S}_{\mathcal{TU}}$ defines a stream by merging two streams of compatible types T and U in the sense that $T \cap U \neq \emptyset$ tuples are valid at the same window interval ω. The Join operator is defined as:

$$S_1 \bowtie^{\omega} S_2\{(e_1 \cdot e_2)_{max(t_1,t_2)} \mid e_1 \in \omega(S_1) \wedge e_2 \in \omega(S_2) \wedge e_1[S_1 \cap S_2] = e_2[S_1 \cap S_2]\} \tag{7}$$

Time-Based Window (Retention Period) The time-based sliding window $\omega_k^T : \mathbb{S}_T \rightarrow \mathbb{S}_T$ takes a stream S and the window size as arguments and defines an output stream by shifting a time interval of size $k > 0$ time units over an input stream.

$$\omega_k(S)\left\{e_{t'} \mid e_{t'} \in S \wedge \exists\, X \subseteq S.X \neq 0 \wedge X = \{e_t \mid e^t \in S \wedge max(t' - k + 1, 0) \leq t \leq t'\}\right\} \tag{8}$$

At a time instant t', the output stream contains all tuples of S whose timestamp intersects with the window of size k that ends at instant t', i.e., the time interval $[max(t' - k + 1, 0), t']$.

Count-Based Window (Retention Period) The count-based sliding window $\omega_n^T : \mathbb{S}_T \rightarrow \mathbb{S}_T$ defines an output stream over time that contains the last $n > 0$ elements over the input stream.

$$\omega_n(S)\left\{e_{t'} \mid e_{t'} \in S \wedge \exists\, X \subseteq S.X \neq 0 \wedge X = \{e_t \mid e_t \in S \wedge t \leq t' \wedge |X| \leq n\}\right\} \tag{9}$$

6.2.4 Language Implementation

The Language Implementation stage comprises the integration of BISEL artefacts resulting from the language design stage in an execution platform. Therefore, necessary transformations of the language meta-model, described during the language design phase, to the respective programming language code are implemented. Our solution was developed using DSL composition techniques. Indeed, as can be seen in Section 3.2 there is already a DSL in literature, BIMSL, developed to resolve queries that combine BIM and Real-time Sensor data. As a result, our goal was to develop BISEL on top of BIMSL and raise the level of abstraction to create a DSL capable of providing to domain experts a query language more familiar and easy to use.

The solution was implemented using the Eclipse Epsilon, a family of languages and tools for code generation, model-to-model transformation, model validation, comparison, migration and refactoring that work out of the box with different types of models such as EMF, UML and XML.

The solution' architecture is depicted in Figure 3 and is composed of three layers, namely *(i)* the data acquisition layer, *(ii)* the data processing layer, and *(iii)* the data presentation layer.

Data Presentation Layer The data presentation layer is the entry point of our system, where all the inter-

actions with domain-users will be performed. Here, domain-users will be able to formalise queries in the BISEL Editor and visualise the queries results through graphical or textual representation.

Data acquisition layer In data acquisition layer Real-time data provided by sensors, BIM models, and other static information are gathered. This static information includes extra information about sensors that BIM models do not support. BIM models in IFC format are kept in a DBMS managed by BIMserver platform which allows to store, maintain and query building and infrastructures information. This data is forwarded to the data processing layer where it will be integrated.

Data Processing layer The data processing layer is divided into two phases, the transformation phase and the query processing phase. In the first one, BISEL queries specified by a domain expert in the graphical editor are transformed into BIMSL queries. Queries results are also managed and transformed to the form of presentation specified by the user in the query. In the second phase, data from the acquisition layer are integrated and processed in BIMSL platform. Because streaming data provided by sensors came in real-time, it has to be processed using a DSMS. The DSMS evaluates complex queries and is also integrated with DBMS to support historical data from sensors and be able to store the static data provided by BIM models. In contrast to streaming data, BIM models in IFC format are stored in the BIMserver platform own database.

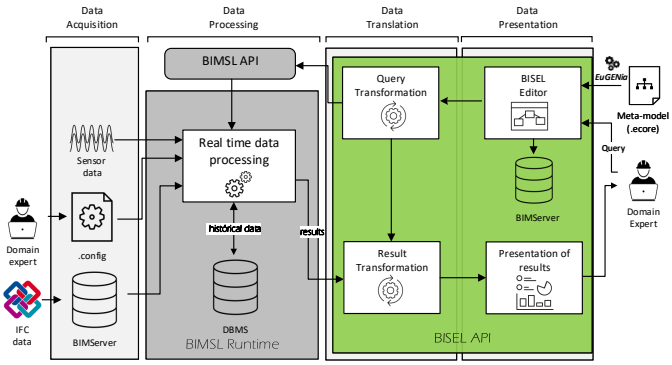

Figure 3: The proposed solution architecture. From left to right: the Data Acquisition, composed of sensor data, IFC files and other static information that may be provided by the user; the Data Processing, where data are integrated, BISEL queries are managed and translated into BIMSL queries; the Data Presentation, where the users can specify queries and several other commands, as well as visualise input measurements and query output results

6.2.5 *BISEL Editor*

The goal was to implement a visual graphical editor for BISEL. Indeed, as previously mentioned, languages with graphical notations are more intuitive and easy to use by non-programmers. Therefore, once the abstract syntax of BISEL is defined using Ecore and the respective EMF code generated, the next challenge was specifying the graphical syntax of the solution using GMF, a powerful application built on EMF for creating graphical editors [30].

To generate the BISEL Editor using GMF, it is necessary to construct three additional models, *(i)* the graph model (GMFGraph), *(ii)* the tooling model (GMFTool), and *(iii)* the mapping model(GMFMap). However, construct these GMF models manually is a tedious and error-prone task, particularly for inexperienced GMF developers [30]. Indeed, it requires developers to handcraft and maintain several low-level, complicated and interconnected iterations between models [30]. To address the highlighted challenge, *EuGENia*, a front-end for GMF, was used to generate the fully functional GMF editor [30, 31]. *EuGENia* reduces GMF complexity and automate the construction of GMF-specific models by merely attaching a few high-level annotations in the Ecore. We then use automated model-to-model and in-place transformations to generate the platform-specific models required by the EMF and GMF code generators in a consistent and repeatable manner.

In our meta-model we use three annotations, namely *@gmf.node*, that applies to classes and denotes that the element should appear on the diagram as a node, *@gmf.links* that applies to non-containment references that should appear in the diagram as edges, and *@gmf.compartment* that denotes that the containment reference will create a compartment where model elements that conform to the type of the reference can be placed. Each annotation accepts different properties to customise the appearance of each component of the graphical editor.

The generated graphical editor displays all language elements in a palette menu and works with a drag-and-drop mechanism. After drag one element to the editor, we can define its properties and the relations with other elements.

6.2.6 *Query Transformations*

The next step in the Language Implementation stage is the Query Transformation, from which BISEL queries are transformed in BIMSL queries to be then evaluated in the BIMSL platform. Since a BISEL query is an instance (model) of our domain model, we were able to use Model Transformations. In our work we used model-to-text transformations. Indeed, BIMSL is a textual language and the model-to-model transformation would require additional effort in modelling the BIMSL language. The transformation was specified using Epsilon Generation Language (EGL), a template-based model-to-text language for generating code, documentation and other textual artefacts from models [29].

6.2.7 *Results transformaion and Presentation of Results*

Along with the query, users are able to specify how the query results will be presented using the table output or graph output element. After receiving the query results from BIMSL platform, the results are managed and transformed according to the type of presentation previously specified by the domain-expert.

7. EVALUATION

To validate if our approach has managed to overcome its requirements and goals, an evaluation of the BISEL language must be performed. In fact, without a proper evaluation, the overall DSL development process is still incomplete.

To support our claims that with our solution we manage

to improve the efficiency, reduce the error rate and have a steep learning curve, we have to perform a complete and unbiased evaluation of our language, comparing to another language that integrates BIM with sensor data in the same conditions. Since BIMSL is our reference model to combine BIM and real-time data, we chose BIMSL to be the alternative language to be used in the usability evaluation of BISEL. In addition, since the target users of BISEL are civil engineers' non-programming domain-experts, we also evaluate the domain-expert satisfaction when using BISEL language.

We start with a description of the evaluation methodology and its participants, followed by an analysis of the results obtained.

7.1 Methodology and Participants

By following an usability evaluation methodology based on literature, it was possible to evaluate the achieved Quality in Use of our DSL. Barišić et al.[6] taking into consideration the ISO/IEC 25010 standard, points out that the most relevant attributes to evaluate the achieved quality in use of DSLs in a real context are the following ones:

- Measuring **effectiveness** means to determine the accuracy and completion when performing queries [6, 7, 8].
- **Efficiency** measurement is related to the level of effectiveness achieved at the expense of various resources, such as mental and physical effort, time, and financial cost.
- When measuring **satisfaction** in use it means freedom from inconveniences and validate how comfortable does the user feel while using the system [4, 6, 7, 8].
- **Accessibility** determines learnability and memorability of language terms.

The BISEL evaluation process is summarised in Figure 4. The whole process starts with a **Subject Recruitment**, followed by the **Task Preparation**, where all evaluation tasks are prepared. The next step is the **Pilot Session**, which is meant to check the training and the evaluation material and then we proceed to the **Evaluation Session** which involves training sessions, exams and final questionnaires for each group. The goal of the Final Questionnaire session is to obtain participants' satisfaction with the language being evaluated and their assessment on accessibility.

Finally, since the overall evaluation approach has been detailed, we are now able to state the goal of our usability evaluation of BISEL. Therefore, the objective of our evaluation experiment is to answer the following research questions:

- (Accessibility) Is the application accessible both for Domain Experts and Software Engineers?
- (Performance) The performance when specifying queries integrating BIM with sensor data increases with BISEL when compared to BIMSL?

Therefore, the subjects to be part of this process. We devise two types of persons involved: non-programmers (non-P) and programmers(P). The non-programmers participants are researchers and civil engineers of Concrete Dams Department and Buildings Department of LNEC and civil engineers of EDP, the dam owner, that are familiar with the infrastructure, but who do not have previous experience on BIM and real-time data processing. The programmers are software developers with a degree in Computer Science and basic knowledge in SQL.

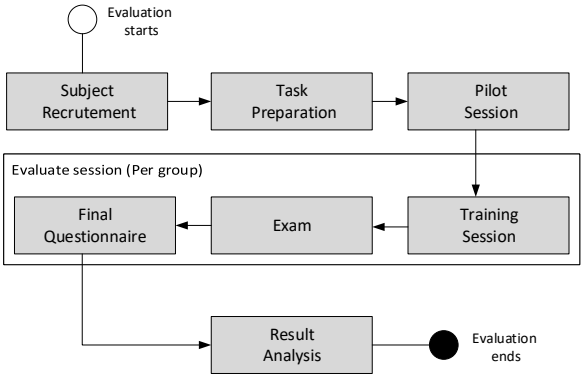

Figure 4: The Activity Diagram representing the evaluation process of the BISEL language.

7.2 Results

The obtained results of our usability evaluation experiment allowed to draw several conclusions concerning each attribute:

- Regarding **Effectiveness**, we have a clear evidence that the solution provides the user with a tool that is very accurate, even for users without previous knowledge in query languages. Additionally, the group of programmers make fewer mistakes specifying queries using BISEL when compared to the alternative.
- Regarding **Efficiency**, the results show that although non-programmers users require more training time and generally spend more to achieve the same goal when compared to programmers, the differences are not significant. In addition, the programmers efficiency results shows that the time spent in specify queries using BISEL require lower mental and physical effort when compared to the time spent using BIMSL.
- Regarding **Satisfaction**, the self-assessment performed by the users allowed to confirm their positive attitude towards BISEL, even considering that BISEL was a language that the users had just learned. The group of programmers presented slightly better satisfaction results when specifying queries using BISEL than when using BIMSL.
- Regarding **accessibility**, the self-assessment performed by the users allowed to confirm that the solution is easy to use, learn and understand for both groups.

At the end of the questionnaires, the users had the possibility of suggesting improvements or making comments. The experts demonstrated very enthusiastic about the solution and made some suggestions for the future practical use of the languages, such as *(i)* a query history mechanism and *(ii)* creation of direct interactions with BIM models in Revit.

Our usability evaluation experiment with the target users of BISEL, domain experts, allowed to confirm that the language meets its requirements. Since the analysis of results revealed successful outcomes, we have determined that, by using BISEL, users without previous knowledge in query

languages can specify queries integrating BIM and real-time sensor data efficiently and effectively. Including Software Engineers in the language assessment, we could also confirm that the rise in abstractions of the languages positively influences the productivity of users in the query specification when compared to the alternative.

8. CONCLUSIONS

The integration of sensors with spacial and other physical and functional information extracted from digital BIM models has been hailed as a means to improve the performance analysis, monitoring, and the maintenance of the built environment. Despite the variety approaches that have recently been proposed by the research community to integrate BIM and sensor data, none has been able to be simultaneously: *(i)* easily adaptable to different application domains, *(ii)* easily used by domain experts and flexible for carrying out all kinds of performance operations, and *(iii)* adaptable to changes without modifying the application code. This thesis explores the possibility of creating a solution capable of dealing with all the challenges mentioned above and therefore, offering to domain experts an empowerment tool that facilitates the integration of real-time sensor data with data extracted from BIM models.

The approach followed consists of the development of a DSL (named BISEL) aiming at simplifying the creation of applications oriented to non-programmers, allowing them to work in a clean, custom, and productive environment with concepts closely aligned with the domain in which they work. One merit of our solution is that the agreement on the DSL abstractions and the key requirements was obtained together with domain experts through Focus Group meetings, which also helped us to understand the solution goals quickly and to plan a case study.

Secondly, BISEL was implemented using a pure model-driven approach through EMF, which comprises code generation facilities to increase productivity and reduce the design and implementation errors. Indeed, by merely describing the language meta-model we were able to generate the programming code of the solution in one-click, and therefore we had much less system to write and could focus on providing better functionalities to fulfil domain experts needs. Using EMF support for model transformations we also were able to reuse an existing DSL previously proposed in literature for integrating BIM and sensor data, BIMSL. Therefore, queries expressed using BISEL are transformed into BIMSL queries using modelling transformations and will then evaluated in the BIMSL platform. Using this technique of composition of DSL's we do not have to develop the solution from scratch. Finally, by integrating EMF with GMF, it was also possible to generate a graphical editor for the solution by simply annotating the language meta-model using *EuGENia* annotations.

To evaluate the language BISEL, we followed a rigorous usability evaluation experiment based on literature. We assessed the user effectiveness, efficiency and satisfaction when specifying queries using BISEL, and language accessibility through a questionnaire. This evaluation experiment allowed to prove that BISEL is easy to use and learn by non-programmers that are also domain experts. Additionally, we also included Software Engineers in the evaluation this offering another level of validation. Indeed, we also demonstrated that BISEL is less error-prone than using the base language directly and, moreover, that using BISEL, a user can specify queries that integrate BIM with sensor data with more confidence.

REFERENCES

[1] A. Albreshne and J. Pasquier. A domain specific language for high-level process control programming in smart buildings. *Procedia Computer Science*, 63 (Euspn):65–73, 2015.

[2] M. Alves, P. Carreira, and A. A. Costa. BIMSL: A generic approach to the integration of building information models with real-time sensor data. *Automation in Construction*, 84(November 2016):304–314, 2017.

[3] M. B. Alves. BIMSL : A Domain Specific Language for Integrating Building Information Models with Sensor Data. (November), 2016.

[4] V. Amaral and M. Goulão. Computer Languages , Systems & Structures Usability driven DSL development with USE-ME. *Computer Languages, Systems and Structures*, 51:118–157, 2018.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002.

[6] A. Barišić, V. Amaral, M. Goulão, and B. Barroca. Quality in Use of DSLs: Current Evaluation Methods. *Proceedings of the 3rd INForum - Simpósio de Informática (INForum2011)*, 2011.

[7] A. Barišić, V. Amaral, M. Goulão, and B. Barroca. Quality in Use of Domain Specific Languages: a Case Study. *Proceedings of the 3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU) at SPLASH,*, pages 65–72, 2011.

[8] A. Barišić, V. Amaral, M. Goulão, and B. Barroca. Quality in Use of DSLs: Current Evaluation Methods. In *Proceedings of the 3rd INForum – Simpósio de Informática (INForum2011)*, 2011.

[9] A. Barišić, V. Amaral, M. Goulão, and B. Barroca. Evaluating the Usability of Domain-Specific Languages. *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*, pages 386–407, 2012.

[10] F. G. Brundu, E. Patti, A. Osello, M. D. Giudice, N. Rapetti, A. Krylovskiy, M. Jahn, V. Verda, E. Guelpa, L. Rietto, and A. Acquaviva. IoT software infrastructure for energy management and simulation in smart cities. *IEEE Transactions on Industrial Informatics*, 13(2):832–840, 2017.

[11] D. Carney, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, and S. Zdonik. Monitoring Streams – A New Class of Data Management Applications. *Vldb*, pages 215–226, 2002.

[12] S. Chakravarthy and Q. Jiang. *Stream Data Processing: A Quality of Service Perspective*, volume 36. 2009.

[13] J. Chen, T. Bulbul, J. Taylor, and G. Olgun. A Case Study of Embedding Real-time Infrastructure Sensor Data to BIM. *Construction Research Congress 2014*, pages 269–278, 2014.

[14] K. Chen, W. Lu, Y. Peng, S. Rowlinson, and G. Q. Huang. Bridging BIM and building: From a literature review to an integrated conceptual framework. *Inter-*

national Journal of Project Management, 33(6):1405–1416, 2015.

[15] M. Y. Cheng, K. C. Chiu, Y. M. Hsieh, I. T. Yang, J. S. Chou, and Y. W. Wu. BIM integrated smart monitoring technique for building fire prevention and disaster relief. *Automation in Construction*, 84(August):14–30, 2017.

[16] J. Choi, J. Shin, M. Kim, and I. Kim. Development of openBIM-based energy analysis software to improve the interoperability of energy performance assessment. *Automation in Construction*, 72:52–64, 2016.

[17] A. L. Ciribini, D. Pasini, L. C. Tagliabue, M. Manfren, B. Daniotti, S. Rinaldi, and E. De Angelis. Tracking Users' Behaviors through Real-time Information in BIMs: Workflow for Interconnection in the Brescia Smart Campus Demonstrator. *Procedia Engineering*, 180:1484–1494, 2017.

[18] T. Clark, A. Evans, P. Sammut, and J. Willans. Applied Metamodelling.

[19] I. Dejanović, M. Tumbas, G. Milosavljević, and B. Perišić. Comparison of textual and visual notations of DOMMlite domain-specific language. *CEUR Workshop Proceedings*, 639:131–136, 2010.

[20] M. Emerson and J. Sztipanovits. Metamodeling Languages and Metaprogrammable Tools. *Handbook of Real-Time and Embedded Systems*, (Mic):1–17, 2008.

[21] Y. Fang, Y. K. Cho, S. Zhang, and E. Perez. Case Study of BIM and Cloud–Enabled Real-Time RFID Indoor Localization for Construction Management Applications. *Journal of Construction Engineering and Management*, 142(7):05016003, 2016.

[22] M. Fowler. Language workbenches: The killer-app for domain specific languages. *Accessed online from: http://www. martinfowler. com/articles/languageWorkbench. html*, pages 1–27, 2005.

[23] J. Gama and P. P. Rodrigues. *Data Stream Processing*. 2007.

[24] K. Garmer, J. Ylvén, and I. C. A. Karlsson. User participation in requirements elicitation comparing focus group interviews and usability tests for eliciting usability requirements for medical equipment: A case study. *International Journal of Industrial Ergonomics*, 33(2): 85–98, 2004.

[25] L. Golab and M. T. Özsu. Issues in Data Stream Management. *SIGMOD Rec.*, 32(2):5–14, 2003.

[26] S. Howell, Y. Rezgui, and T. Beach. Integrating building and urban semantics to empower smart water solutions. *Automation in Construction*, 81:434–448, 2017.

[27] B. Jean. mocap Data types - XDIF. pages 171–188, 2005.

[28] J. Kitzinger and R. Barbour. *Developing Focus Group Research: Politics, Theory and Practice*. SAGE Publications, 1999. Available at `https://books.google.pt/books?id=fyvOGT2Ao3MC` (visited in Jan. of 2018).

[29] D. Kolovos, L. Rose, R. Paige, and A. Garcia-Dominguez. *The Epsilon Book*. Eclipse, 2010.

[30] D. S. Kolovos, L. M. Rose, S. bin Abid, R. F. Paige, F. A. C. Polack, and G. Botterweck. Taming EMF and GMF Using Model Transformation. *Proc. of the 13th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS 2010)*, pages 211–225, 2010.

[31] D. S. Kolovos, A. García-Domínguez, L. M. Rose, and R. F. Paige. Eugenia: towards disciplined and automated development of GMF-based graphical model editors. *Software and Systems Modeling*, 16(1):229–255, 2017.

[32] J. K. Lee. Building Environment Rule and Analysis (BERA) Language and its Application for Evaluating Building Circulation and Spatial Program. *Georgia Tech University*, page 197, 2011.

[33] N. Li, B. Becerik-Gerber, B. Krishnamachari, and L. Soibelman. A BIM centered indoor localization algorithm to support building fire emergency response operations. *Automation in Construction*, 42:78–89, 2014.

[34] W. Mazairac and J. Beetz. BIMQL - An open query language for building information models. *Advanced Engineering Informatics*, 27(4):444–456, 2013.

[35] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005.

[36] S. Niu, W. Pan, and Y. Zhao. A BIM-GIS Integrated Web-based Visualization System for Low Energy Building Design. *Procedia Engineering*, 121:2184–2192, 2015.

[37] J. W. Park, J. Chen, and Y. K. Cho. Self-corrective knowledge-based hybrid tracking system using BIM and multimodal sensors. *Advanced Engineering Informatics*, 32:126–138, 2017.

[38] D. Pasini, S. M. Ventura, S. Rinaldi, P. Bellagente, A. Flammini, and A. L. C. Ciribini. Exploiting Internet of Things and building information modeling framework for management of cognitive buildings. *2016 IEEE International Smart Cities Conference (ISC2)*, 40545387(40545387):1–6, 2016.

[39] J. B. Piers, G. Hillairet, F. Jouault, I. Kurtev, and William. Bridging the MS/DSL Tools and the Eclipse Modeling Framework. *In: Proceedings of the International Workshop on Software Factories at OOPSLA 2005*, (Mic), 2005. Available at `http://www.softwarefactories.com/workshops/OOPSLA-2005/Papers/Bezivin.pdf` (visited in Sept. of 2017).

[40] Z. Riaz, D. J. Edwards, E. A. Parn, C. Shen, and F. Pena-Mora. BIM and sensor-based data management system for construction safety monitoring. *Journal of Engineering, Design and Technology*, (October):00–00, 2017.

[41] A. Svenfelt, R. Engstrom, and O. Svane. Decreasing energy use in buildings by 50% by 2050 - A backcasting study using stakeholder groups. *Technological Forecasting and Social Change*, 78(5):785–796, 2011.

[42] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. Kats, E. Visser, and G. Wachsmuth. DSL Engineering Designing, Implementing and Using Domain-Specific Languages. page 558, 2013. Available at `http://dslbook.org` (visited in Sept. of 2017).

[43] X. Wang. *BIM Handbook: A guide to Building Information Modeling for owners, managers, designers, engineers and contractors*, volume 12. 2012. Available at `https://epress.lib.uts.edu.au/journals/index.php/AJCEB/article/view/2749` (visited in Sept. of 2017).

[44] S. Wilkinson. Focus group methodology: A review. *International Journal of Social Research Methodology*, 1(3):181–203, 1998.

[45] J. Zhang, B.-C. Seet, and T. Lie. Building Information Modelling for Smart Built Environments. *Buildings*, 5 (1):100–115, 2015.