Applied Computer Systems

doi: 10.1515/acss-2015-0010

2015/17

Towards a Standard-Based Domain-Specific Platform to Describe Points of Interest

Vicente García-Díaz [1], Jordán Pascual Espada [2], B. Cristina Pelayo García-Bustelo [3], Juan Manuel Cueva Lovelle [4], Janis Osis [5],
[1, 2, 3, 4] *University of Oviedo, Spain,* [5] Riga *Technical University, Latvia*

***Abstract* – With the proliferation of mobile and distributed systems capable of providing its geoposition and even the geoposition of any other element, commonly called point of interest, developers have created a multitude of new software applications. For this purpose, different technologies such as the GPS or mobile networks are used. There are different languages or formats used to define these points of interest and some applications that facilitate such work. However, there is no globally accepted standard language, which complicates the intercommunication, portability and re-usability of the definitions of points of interest currently in use. In this paper, we take the first steps towards a language and a development environment independent of the underlying technologies, allowing developers to define the points of interest in a simple and fast way, and automatically generate other different formats from the same definition that can be considered a bridge among current technologies. We use the Model-Driven Engineering approach, focusing on the creation of models to abstract the definition of systems from the underlying technologies.**

***Keywords* – Augmented reality, development environment, domain-specific language, point of interest.**

I. Introduction

A Point of Interest (POI) is a specific point location that someone may find useful or interesting for something in a concrete Region of Interest (ROI), typically grouped by POI collections. The concept of POI is widely used in different fields, such as medicine or environment exploration. Thus, for example, Chan et al. [1] refer to POIs in a human body that were traced on image frames, allowing different pathologies to be measured, or Erdelj et al. [2] combine coverage of multiple POIs and network connectivity preservation with environment exploration in order to capture dynamic processes in a specific monitored area.

New technologies have allowed the development of multiple cartography software applications mostly focusing on Global Positioning System (GPS) navigation software [3] or, more generally, Geographic Information System (GIS) [4]. They are based on information that can be located spatially (POIs) and represented as a latitude, longitude and altitude triple, providing operations on and between different positions. In addition, Augmented Reality (AR) applications [5] use the same principles to show information to users depending on their current position at a specific moment of time. Some common POIs include street names and numbers, gas stations, hospitals, restaurants, tagged photos taken with digital cameras, GPS-tracked vehicles that are moving etc.

Given the importance of POIs, especially from the geographical point of view, large databases of information are being created, for example, the OpenStreetMap Project, consisting of a knowledge base that creates and distributes user-generated geographic data for the world [6]. However, one of the major problems encountered when working with POIs is the heterogeneity that exists in the formats used to represent them. There are several reasons that led to this situation among which may include: 1) a large number of different software tools that work with points of interest; 2) the lack of a standard widely adopted in the industry; and 3) the rapid expansion in a short time span, which has caused different companies working simultaneously and with similar objectives, but taking different approaches. Thus, tools are incompatible with each other and developers spend many resources trying to convert information to other format types by ad-hoc solutions.

Although there is still no comprehensive solution to avoid the heterogeneity of formats used by different manufacturers, there are solutions that can help to reduce the impact of this situation. Thus, different tools and software development approaches continuously appear in the software engineering field, trying to abstract the development from specific platforms or technologies (e.g., virtual machines, APIs, frameworks etc.). It is widely considered that the Model-Driven Engineering (MDE) approach, in which the level of abstraction of developments is increased through the use of models, is a step forward in the development of software [7], since developments are being benefited from the advantages provided by MDE (e.g., in García-Díaz et al. [8] food traceability systems for different clients are created in a quick and dynamic way).

MDE is based on the use of models, which conform to a single domain-based metamodel, which in turn is defined based on a common meta-metamodel, root of all the elements of any software development. This idea makes up the architecture of four layers defined in the Model-Driven Architecture (MDA) standard [9]. The common base allows for a wide range of supported environments and tools working together. As a result, if a metamodel for a specific knowledge domain is defined (e.g., food traceability or points of interest), it would be possible to create a Domain-Specific Language (DSL) [10] based on MDE tools, designed only to define the important specific items (e.g., food manufacturing processes or features of points of interest). Internally, the use of standard-based modeling technologies allows direct and automatic transformations to different formats defined by different software manufacturers. There are many studies in MDE that serve to advance in the systematic use of DSLs. For example, the authors in [11] take advantage of use cases to provide a formal base for generating standard-based models or the authors in [12] work on formal trace links to avoid inconsistency between software and specifications.

The main aim of this paper is to take the first steps towards the creation of a standard-based platform for defining points of interest in a simple and common way. Internally, definitions are automatically transformed into different formats. Thus, specific goals are:

1. To identify the basic elements that a format/representation language for POIs must possess;
2. To create a DSL to define POIs. We call it PoiDSL;
3. To allow automatic transformation of definitions made with PoiDSL to any other format;
4. To provide an Integrated Development Environment (IDE) to work with PoiDSL. We call it PoiIDE;
5. To study the advantages of the proposal by a comparison with other alternatives.

The remainder of this paper is structured as follows: in Section II, we present a description of the relevant state of the art (goal [1]); in Section III, we describe our proposal (goals [2]–[4]); in Section IV, we discuss a comparison of the proposal with some alternatives and a qualitative analysis (goal [5]) and finally, in Section V, we indicate our conclusions and future research to be conducted.

II. Background

There are many formats used to represent points on the Earth. The simplest ones are based on plain text and Comma-Separated Values (CSVs) data, which are too generic to promote the automatic and semantic processing of contents, having their strength in writing tabular data. However, they have important drawbacks such as the inability to organize information hierarchically or perform validations using a schema or metamodel.

The GPS eXchange format (GPX)[1] is an XML data format for the interchange of GPS data between applications and Web services. GPX allows the definition of collections of POIs. If the collection is an ordered list of points describing a path, it is called track. If, in addition, the track is used to lead to a destination, it is called route. Since this format does not include a large amount of information for each POI, some companies created extensions to the GPX format for including more information such as street addresses, phone numbers, business categories, air temperature, depth of water etc.

The Overlay format (OV2) [13] contains a database of POIs that are typically used by TomTom GPS devices. An OV2 document consists of a sequence of variable-length records with elements of a predefined number of bytes. In addition to this simple format, it is possible to supplement the information with external itinerary files created in plain ASCII [14]. They consist of lines, each marking one point in the itinerary. TomTom needs the use of other proprietary external file formats, such as CAP to extend the location-sensitive menu, TMF to change the icons of elements on the screen or GF to provide graphic information regarding shapes and elements.

The OpenStreetMap project creates and distributes free geographic data using the OSM format [6]. The basic concepts on the OpenStreetMap world are points, which can be used to represent POIs, such as shops or fuel stations, areas, which are more detailed than points, since they provide information on the boundaries of the elements and lines that are used to represent elements, such as roads or rivers. The OSM file format has many other variations, such as Protocolbuffer Binary Format (PFB) that is intended as an alternative to the XML format; the O5M format that was designed to be a compromise between OSM and PBF formats; or OSM JSON that is a JavaScript Object Notation (JSON) [15] variant of OSM XML.

The Keyhole Markup Language (KML) [16] works on many applications, such as Google Earth, Google Maps, NASAWorld-Wind or ESRI ArcGIS Explorer. It is intended to show any kind of geographic data (POIs, descriptions, ground overlays, paths and polygons). POIs are represented through the Placemark construction. In turn, a PlaceMark is a Feature element with associated Geometry, which inherits from Object, the root element of any KML document. It is maintained by the Open Geospatial Consortium[2] (OGC). In addition, it provides a mechanism for extensions used by Google for creating specific fields for the Google Earth application (e.g., the flyTo element to specify a point in space to which the browser will fly at a given moment). Another interesting format is KARML [17], an extension of KML intended to include AR-specific content.

The Augmented Reality Markup Language (ARML) [18] is intended to focus on AR applications by adding specific information not presented in other formats like KML and removing information that may not be interesting from the point of view of AR. In its version 1.0, it is a proprietary format used mainly in the AR browser called Wikitude[3]. Version 2.0 is a proposed standard that is currently being reviewed by the Open Geospatial Consortium[4]. In addition to Wikitude, there are many other augmented-reality browsers like Junaio or Layar [19]. They are applications which may, among other things, display information about the user surroundings, usually in a mobile camera view. Fig. 1 is a screenshot of a simple AR application created for the Android operating system. It uses the ARML format to retrieve the information of the points of interest closest to the user.

[1] http://www.topografix.com/gpx.asp, accessed in Nov. 2014

[2] http://www.opengeospatial.org, accessed in Nov. 2014

[3] http://www.wikitude.com, accessed in Dec. 2014

[4] http://www.opengeospatial.org/projects/groups/arml2.0swg, accessed in Dec. 2014

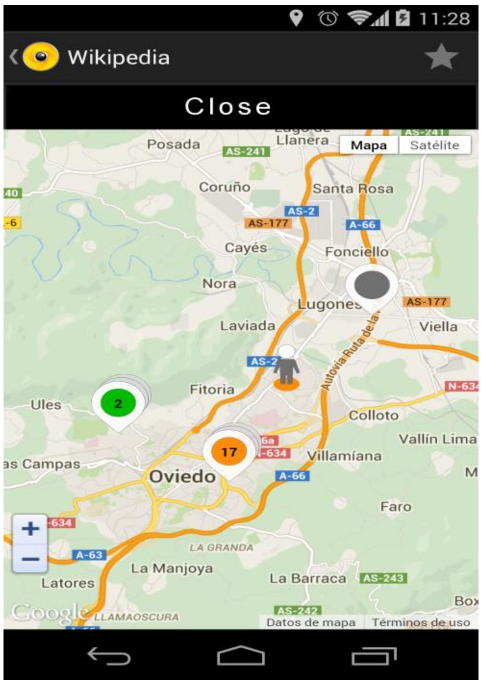


Fig. 1. Augmented reality browser example.

To create POIs, the Junaio AR browser proposes the AREL language, which is composed of two parts. AREL XML, a XML-based language, which defines the static information of all the content and AREL JavaScript, which is an API to define the interactions and behaviors of elements. The concept is similar to other systems, such as Wikitude that, in addition to using a format to define the static part of the points of interest, uses a General-Purpose Programming Language (GPL) to define behaviors. The main difference is the possibility of being combined and integrated in Junaio.

Finally, there are other formats, for example, Garmin Mapsource (GDB), Pocket Street Pushpins (PSP), Maptech Marks (MSF), Maptech Waypoint (MXF) or OziExplorer (WPT) that may not be as well-known as the format listed above and that they share virtually the same data, so we do not go into more detail in this paper.

With all this variety of formats, there are several tools that facilitate the creation, edition and transformation of POIs. For example, PoiEdit[5] supports reading and writing in a variety of formats by displaying items on a map view. PoiEditor[6] is a web-based tool that allows the creation of lists of POIs in a simple way. There are even other tools with similar purposes such as ExtraPoiEdit[7] or PoiExpert[8]. In addition, there are lots of custom applications created for making conversion among specific formats (e.g., OV2 to KML, OV2 to CSV, KML to GPX etc.). Although there are very useful tools, they are limited because depending on the case different reasons include: 1) closed architectures that do not allow extensions; 2) small number of formats that are handled; 3) small number of features for POIs that can be defined; 4) lack of standards to facilitate interoperability with other environments and the creation of extension points; and 5) lack of a flexible environment to define POIs in a platform-independent way.

A. Basic Elements for Specifying Points of Interest

Table I shows a list of features that are commonly used to describe POIs through formats supported by different tools. For example, the ARML format manages all of such features. It is important to note that not all the formats or languages provide all the features, and some of them support more features than the included ones in its language schema. As an example, the Wikitude software development kit (SDK) allows creating very complex and rich AR worlds with 3D models, animations, sounds and videos not supported in ARML 1.0, its native AR definition format, but using the JavaScript programming language. Indeed, other languages to define POIs such as AREL manage general metadata information (e.g., copyright, creator, etc.) also not included in ARML 1.0.

Among all the elements used to describe POIs, in this paper we focus, as a first step, on structural information supported by ARML 1.0, arguably the most important language to define POIs focusing on AR applications, our first target domain of knowledge. Other features defined by other formats, such as the definition of custom views, 3D models, attachments, positioning through natural features or behavioral information, are beyond the scope of this paper.

TABLE I

COMMON ELEMENTS USED TO DESCRIBE POINTS OF INTEREST FOR DIFFERENT FORMATS

	Description
Point of Interest	
Id	A unique identifier for the POI
Name	Name of the POI that is displayed as POI title
Description	A brief description of the POI
Latitude	Latitude of the POI
Longitude	Longitude of the POI
Altitude	Altitude of the POI
URL	URL associated to the POI
Phone	Phone number associated to the POI
Email	Email address associated to the POI
Address	Physical address associated to the POI
Icon	URL of an icon associated to the POI
Provider	
Id	A unique identifier for the provider
Name	The name of the provider
Description	A brief description of the provider of the POI
URL	A URL that represents the provider
Logo	The URL of a corporate image

[5] http://www.poiedit.com, accessed in Nov. 2014

[6] http://www.poieditor.com, accessed in Nov. 2014

[7] http://garmin.gps-data-team.com/extra, accessed in Nov. 2014

[8] http://tomtom.gps-data-team.com/poi/poi-expert.php, accessed in Nov. 2014

III. Overview of the System

Decision in favor of a new DSL is usually not easy because it is much less expensive (both in time and cost) to adopt an existing DSL if available or even to use a General-Purpose Languages (GPL), such as Java or C#. There are mainly only two reasons why it is worth creating a new DSL [20]: 1) improved software economics, giving some authors as a reference point three developments [21] to obtain a positive return on investment; and 2) easy-to-use, i.e., people with less domain and programming expertise are able to develop software, and even end-users with some domain, but no programming expertise [22], [23].

Since our goals are compatible with both criteria, we have created a new DSL based on common elements used to describe the structural part of any POI. To that end, we have used the Xtext framework [24], which allows the creation of both GPLs and DSLs in a relatively easy way [25]. From a grammar and some other definitions, it is possible, for example, to get a working parser and linker and also a complete Eclipse-based IDE [26].

A. PoiDSL

Next, there is a fragment of the context-free Xtext grammar used as a basis of the PoiDSL language. It is possible to see the definition of each POI through the use of three grammar rules.

```
Poi:
'Poi' '{'
'id' '=' name = ID
'provider' '=' provider = [Provider]
'name' '=' realName = STRING
('description' '=' description = STRING)?
'location' '=' location = Location
('infoURL' '=' infoURL = STRING)?
('iconURL' '=' iconURL = STRING)?
('phone' '=' phone = STRING)?
('email' '=' email = STRING)?
('address' '=' address = STRING)?
'}' ;

Float: ('-')?INT ('.'INT)?;
Location: Float ',' Float (',' Float)?;
```

Note that there are some elements such as phone numbers, email addresses or URLs that are difficult to validate using grammar expressions since that is an error-prone task. This is the reason why we try to leave the grammar as simple as possible (e.g., we just indicate that an email address should have a String type) and we rely on third-party libraries to make the necessary validations.

For example, we use the libphonenumber library[9] to check whether phone numbers are correct. The Apache Commons project[10] is used to check whether URLs are well formed. In addition, we make use of the JavaMail API[11] that provides a platform-independent framework to build mail and messaging applications to check whether email addresses are correct.

```
@Check
def checkValidEmailAddress(Poi poi) {
  if (emailAddr != null)
      try {
        emailAddr = new
        InternetAddress(poi.email);
        emailAddr.validate();
      } catch (AddressException ex) {
        error('The email address is not
        valid. Please change it.',
        PoiDSLPackage.Literals.POI__EMAIL,
        INVALID_EMAIL);
      }
  }
```

The code above shows what is needed in Xtext to check if each of the email addresses typed by the developers is valid. Only a few lines of code are required to indicate whether the development environment should display an error message to the developer explaining why it occurs and the exact point in the code. Similarly, it might display a warning message instead of an error one if the issue were not so serious.

The Xtext-based grammar is transformed internally and automatically into an ANTLR grammar [27] to implement the lexer (lexical analysis) and the parser (syntactic analysis) that is used when a programming language is being defined. In addition, it also generates all the necessary infrastructure to create the Abstract Syntax Tree (AST) to perform a semantic analysis on the language elements [28]. The iteration through the tree is performed using model-based technologies, particularly the Eclipse Modeling Framework [29], which serves to ensure interoperability of the generated DSL with many other model-based existing tools, such as the tools defined in the Eclipse Modeling Project [30], to help improve software development productivity.

B. Transformation from PoiDSL to Other Formats

The fragment of code below shows a fragment of the template that is used to generate artifacts from any of the models defined using PoiDSL. In this example, the generation is focused on automatically converting the model into an ARML file (particularly, providers and POIs). The idea of this approach is to generate from a model, easily and automatically, the code for the points of interest in any other format (it would only be necessary to add new templates). Similarly, it could be possible to perform the opposite operation, which is, reusing the PoiDSL metamodel to create models from other formats. That would be a key step to benefit from all the advantages of integration and reuse offered by the MDE approach (e.g., the use of common repositories and version control systems for models).

[9] https://code.google.com/p/libphonenumber, accessed in Dec. 2014

[10] http://commons.apache.org, accessed in Dec. 2014

[11] http://www.oracle.com/technetwork/java/javamail, accessed in Dec. 2014

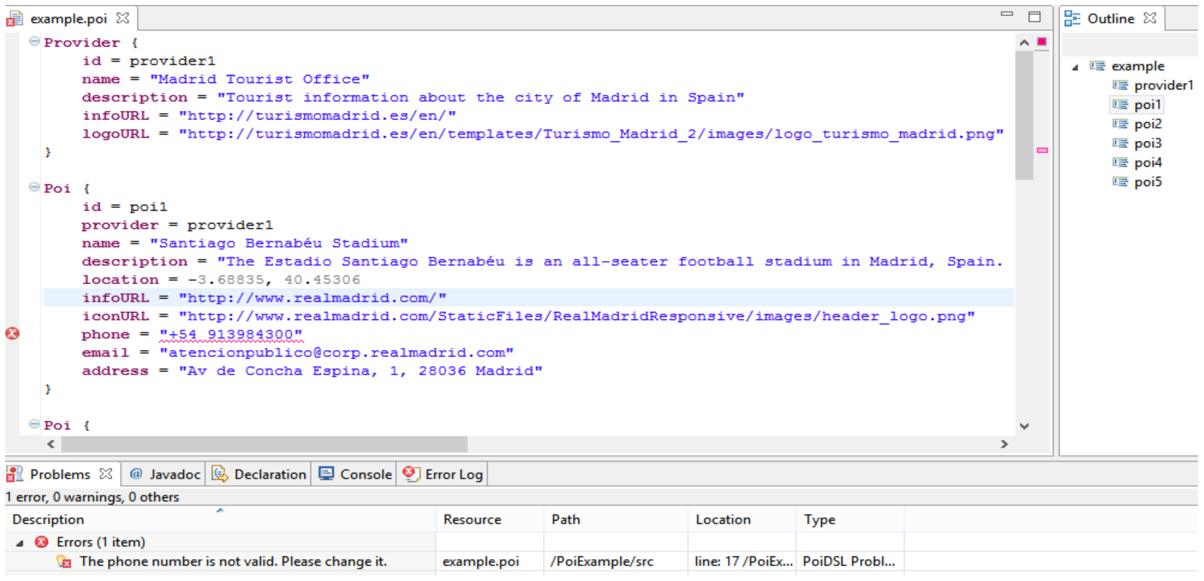
```
def compile(Provider p) '''
    <ar:provider id="«p.name»">
      <ar:name>«p.realName»</ar:name>
      «IF p.description != null»

  <ar:description>«p.description»</ar:description>
      «ENDIF»
      «IF p.infoURL != null»
  <wikitude:providerUrl>«p.infoURL»</wikitude:provid
erUrl>
      «ENDIF»
      «IF p.logoURL != null»
      <wikitude:logo>«p.logoURL»</wikitude:logo>
      «ENDIF»
    </ar:provider>
  '''
  def compile(Poi p) '''
    <Placemark id="«p.name»">
      <ar:provider>«p.provider.name»</ar:provider>
      <name>«p.realName»</name>
      «IF p.description != null»
      <description>«p.description»</description>
      «ENDIF»
      <wikitude:info>
        «IF p.iconURL != null»

  <wikitude:thumbnail>«p.iconURL»</wikitude:thumbnai
l>
        «ENDIF»
        «IF p.phone != null»
        <wikitude:phone>«p.phone»</wikitude:phone>
        «ENDIF»
        «IF p.infoURL != null»
        <wikitude:url>«p.infoURL»</wikitude:url>
        «ENDIF»
        «IF p.email != null»
        <wikitude:email>«p.email»</wikitude:email>
        «ENDIF»
        «IF p.address != null»

  <wikitude:address>«p.address»</wikitude:address>
        «ENDIF»
      </wikitude:info>
      <Point>
        <coordinates>«p.location»</coordinates>
      </Point>
    </Placemark>
    '''
```

C. PoiIDE

Based on the Xtext architecture, some of the features included in development environment called PoiDSL are:

- Custom syntax-highlighting to distinguish the different elements of the language (e.g., keywords, comments or variables). This is done by implementing the Xtext interfaces IHighlightingConfiguration and ISemanticHighlightingCalculator;
- Content assistant to help the developer to write a code faster and more efficiently through the use of the auto-complete functionality (extending the TerminalsProposalProvider class);
- Static validation of the language elements to detect syntactic and semantic issues (extending the AbstractDeclarativeValidator class);
- Suggestions for fixing errors or problems identified in the code (extending the DefaultQuickfixProvider class);
- Templates that allow developers to reduce the learning curve for typical operations;
- Formatting the code through a feature called code beautifier to distribute it properly and promote its maintenance (extending the AbstractDeclarativeFormatter class);
- Outline view fully configurable to both the elements that appear and text or icons attached to them (extending the DefaultObjectLabelProvider class).

Fig. 2 is a screenshot of the environment when a model is being created. It shows different features. For example, the syntax-highlighting for different elements (e.g., keywords, strings, codes, numbers), the static validation marking a phone number as not valid because settings state that phone numbers must be valid in Spain (of course, we could customize the settings) and the outline view showing a summary of the elements that are being used.



```
Provider {
     id = provider1
     name = "Madrid Tourist Office"
     description = "Tourist information about the city of Madrid in Spain"
     infoURL = "http://turismomadrid.es/en/"
     logoURL = "http://turismomadrid.es/en/templates/Turismo_Madrid_2/images/logo_turismo_madrid.png"
}

Poi {
     id = poi1
     provider = provider1
     name = "Santiago Bernabéu Stadium"
     description = "The Estadio Santiago Bernabéu is an all-seater football stadium in Madrid, Spain.
     location = -3.68835, 40.45306
     infoURL = "http://www.realmadrid.com/"
     iconURL = "http://www.realmadrid.com/StaticFiles/RealMadridResponsive/images/header_logo.png"
     phone = "+54 913984300"
     email = "atencionpublico@corp.realmadrid.com"
     address = "Av de Concha Espina, 1, 28036 Madrid"
}

Poi {
```

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
The phone number is not valid. Please change it.	example.poi	/PoiExample/src	line: 17 /PoiEx...	PoiDSL Probl...

Fig. 2. Augmented reality browser example.

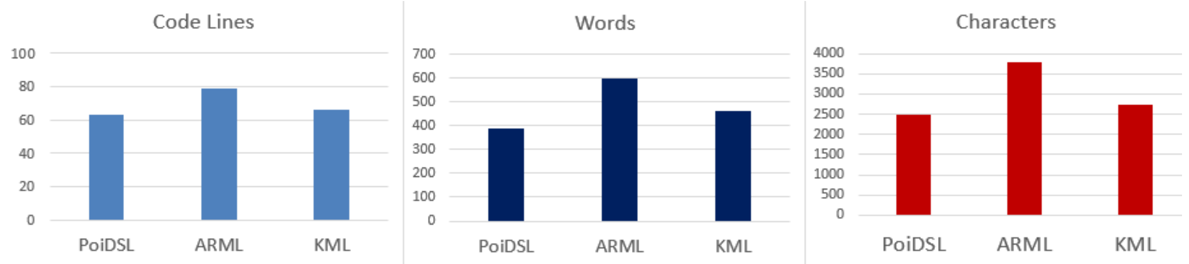


Fig. 3. Comparing PoiDSL, ARML and KML working with one provider and five POIs.

In addition, it is possible to perform other customizations such as specifying the scope of the variables of the language. Thus, the PoiDSL is a full-fledged development environment integrated in the Eclipse platform with the resulting advantages it provides (e.g., well-known and proven platform for developers, large number of tools and plug-ins, open environment etc.).

IV. Evaluation

The sections below are dedicated to a qualitative and quantitative study to show the characteristics of PoiIDE and PoiDSL, justifying the design and the need for their creation.

A. Qualitative Analysis

To achieve a better quality of the language and the environment design and a better acceptance among its users, Karsai et al. [31] have proposed some guidelines largely based on their experience in developing languages as well as relying on existing guidelines on programming and modeling languages. Table II serves to verify that these guidelines are met.

B. Quantitative Analysis

In this section we evaluate the PoiDSL language. We obtain a quantitative measurement that allows us to evaluate the main objective of our proposal; simplify and make more agile the definition of points of interest in software applications.

In this first step of the development, we are going to do a brief comparison between the definitions of five different POIs through the use of PoiDSL, the standard way used by the ARML format, and the KML format used by companies as Google. Since with PoiDSL it is possible to automatically generate a code for ARML and any other format, if the syntax used by PoiDSL is more compact, then it can clearly be seen as advantageous over other languages or formats. The measured aspects in the code and the structure are the ones below:

- Code lines: it refers to the number of lines of information needed to define the POIs in each case;
- Words: a number of words used;
- Characters: a number of characters, spaces included.

Table III shows the information included in the case study regarding the provider and Table IV shows the information included in the case study regarding the different POIs.

In the obtained results of the analysis (Fig. 3), we can observe that with PoiDSL we require fewer code lines, words and characters to define the same information than with the ARML format. The same applies to KML. Moreover, with the generation of code for the KML format, we lose some information because not all the information defined with PoiDSL is easily supported by KML (e.g., provider information or email addresses). Therefore, the difference would even be more representative in this case.

V. Conclusion and Future Research

In this paper, we have presented the first version of a language for defining POIs (PoiDSL) and a development environment to facilitate working with those POIs (PoiIDE). This has been done by identifying basic elements that are useful to define a geographic point of interest to a person. In addition, we have defined mappings for transforming models made with PoiDSL to ARML and KML, and created the basis to do the same with other different popular formats (e.g., KARML), which favor the development and increase productivity and interoperability among systems. Finally, the use of PoiDSL through the PoiIDE is easier than the manual and specific handling of other formats with identical purposes.

Future research will be aimed at improving and adapting both PoiIDE and PoiDSL with new formats and features to define POIs. Specifically, we will delve into the static part (e.g., custom views, 3D models, attachments, positioning through natural features, styles, meta-information etc.) and emphasize the dynamic part (i.e., behavior). One of the main tasks to be performed will be the adaptation of the environment to the developing standard called ARML 2.0, since its final adoption could guide the future of the industry regarding augmented reality. Another future study will be focused on performing mappings between different formats through a bridge metamodel [37], which serves as a basis to easily make transformations among them (e.g., between ARML and KML and vice versa). All these tasks are fundamentally based on the technologies offered by the MDE [39], [40], [41], specifically the Eclipse Modeling Project that being not tied to any organization, which defines standards for MDE such as the Object Management Group (OMG)[12], implements many of its standards [44].

Finally, we will perform a usability study with real users for quantifying how simple, easy and intuitive our proposal is for them [38]. The idea is to work with people with different profiles and ask them to define several POIs using different techniques. That way, we will observe, among other things, the efficiency, the learning curve and the number of errors made.

[12] http://www.omg.org, accessed in Nov. 2014

TABLE II

GUIDELINES FOR A BETTER QUALITY AND BETTER ACCEPTANCE AMONG ITS USERS

Guideline	**Accomplishment**
Language purpose	
Identify language uses early	The language is used mainly for documentation of knowledge and code generation.
Ask questions about uses	Any person with interest in defining POIs will be able to model with the language.
Make your language consistent [32]	It is consistent with the sole idea of defining and creating POIs.
Language realization	
Decide carefully whether to use graphical or textual realization [42], [43]	Textual realization is based on the advantages noted by Groenniger et al. [33]: 1) less space is needed to display the same information; 2) more efficient creation of code; 3) easier integration with other languages; 4) higher speed and better quality of the formatting; 5) platform and tool independency; and 6) better version control support. Besides, graphical realizations provide a better overview and ease the understanding of models [31], but in a very close and specific domain like POIs, we believe that the advantages of textual languages are more important.
Compose existing languages where possible	Instead of starting from scratch, we have used the entire ecosystem of tools provided by the Eclipse Modeling Project, specifically Xtext, DSL to define other DSLs, which rely heavily on the use of the Xtend language, an extension of the Java language, primarily intended to support the creation of DSLs (e.g., validations and code generation)
Reuse existing language definitions	To create PoiDSL, we have used the core grammar of Xtext as a basis to avoid redefining elements already defined previously.
Reuse existing type systems	Related to the previous point, we have reused the core data types defined by the creators of Xtext with the aim of reusing the existing knowledge.
Language content	
Reflect only the necessary domain concepts	It only contains the basic elements needed to generate a code in the different target formats, so no extra domain concept is added.
Keep it simple	With a small number of elements, simple syntax and reduced domain of knowledge, we think that the language is easier than other alternatives. The quantitative analysis also suggests the same idea.
Avoid unnecessary generality	Due to the close domain of the language, we did not include the generalization concept, meeting with the principle of designing only what is necessary.
Limit the number of language elements	The language is small, having only 13 domain-specific keywords (e.g., Java has 50 generic keywords and C# even more).
Avoid conceptual redundancy	Each fact can only be described in a unique way, avoiding redundancy.
Avoid inefficient language elements	Each element is needed for clarity and used with the only purpose of allowing the generation of the final code, so there are no inefficient language elements.
Concrete syntax	
Adopt existing notations domain experts use [34]	POIs are usually defined with textual notations using a tree-based structure, similar to the structure adopted by most of the formats used commonly to define POIs, based on XML.
Use descriptive notations	The language has a small number of keywords with syntax highlighting and code completion support. In addition, frequently-used symbols in other languages such as =, { or } maintain their semantics.
Make elements distinguishable	Keywords, different syntax highlighting and an outline view are used to make elements distinguishable.
Use syntactic sugar appropriately	We avoid syntactic sugar since we think that in a small DSL expressing the same concepts in different ways can be counterproductive, confusing users and hindering validation and code generation unnecessarily.
Permit comments [35]	Support for common types of comments: single-line comments (//) and multi-line comments (/*..*/).
Provide organizational structures for models	Organizational structures such as packages are important for complex systems. However, to keep the language simple, we intend to have the definition of a collection of the POIs in the same organizational structure.
Balance compactness and comprehensibility	The quantitative analysis shows that this approach may require fewer elements than other approaches. However, since it is a DSL with concrete semantics for the domain, it is even more comprehensible.
Use the same style everywhere	All the elements of the language have the same look-and-feel and we do not embed any external language that can complicate the understanding of the language by using another syntax.
Identify usage conventions	Based on an ANTLR grammar, we define typical usage conventions, including notation of identifiers, order of elements or type of comments.
Abstract syntax	
Align abstract and concrete syntax	We have taken into account the three principles mentioned in Karsai et al. [31]: 1) elements that differ in the concrete syntax also have different abstract notations (e.g., POIs and providers are based on different metaclasses); 2) elements that have a similar meaning can be internally presented by reusing concepts of the abstract syntax (e.g., the Location rule for placing POIs has been created using three Float rules for longitude, latitude and altitude); and 3) the abstract notation should not depend on the context in which an element is used but only on the element itself.
Prefer a layout which does not affect	To simplify the usage of the DSL, the layout of the models does not affect the semantics. For example, modelers

Guideline	Accomplishment
translation from concrete to abstract syntax	can use tabs, spaces or line breaks whenever they want. However, PoiIDE provides the feature called a code beautifier, also provided by some environments to automatically place the language elements in a way easily understandable for most potential users.
Enable modularity [36]	It is possible to decompose the code into smaller files, referencing them from other files. However, for this small language, we think that it is not necessary and it may unnecessarily increase the difficulty of use.
Introduce interfaces	Interfaces are an important feature in complex systems, increasing flexibility and maintenance. However, we do not need them in our DSL because it is a simple declarative language.

TABLE III

PROVIDER USED IN THE CASE STUDY

Parameter	Value
id	provider1
name	Madrid Tourist Office
description	Tourist information about the city of Madrid in Spain
infoURL	http://turismomadrid.es/en/
logoURL	http://turismomadrid.es/en/templates/Turismo Madrid 2/images/logo turismo madrid.png

TABLE IV

POIs USED IN THE CASE STUDY

Parameter	Value
POI 1	
id	poi1
provider	provider1
name	Santiago Bernabéu Stadium
description	The Estadio Santiago Bernabéu is an all-seater football stadium in Madrid, Spain. It was inaugurated on December 14, 1947 and is owned by Real Madrid Club de Fútbol. It has a current capacity of 81,044 spectators.
location	−3.68835, 40.45306
infoURL	http://www.realmadrid.com/
iconURL	http://www.realmadrid.com/StaticFiles/RealMadridResponsive/images/header logo.png
phone	+34 913984300
email	atencionpublico@corp.realmadrid.com
address	Av. de Concha Espina, 1, 28036 Madrid
POI 2	
id	poi2
provider	provider1
name	Parque Warner Madrid
description	Parque Warner Madrid is a theme park located 25 km southeast of Madrid, Spain, in the municipality of San Martín de la Vega.
location	−3.59406, 40.23217
infoURL	http://www.parquewarner.com/
iconURL	http://www.parquewarner.com/sites/all/themes/custom/WAR theme/logo.png
phone	+34 902024100
email	atencionalcliente@parquewarner.com
address	M-301, Km 15.5, 28330 San Martín de la Vega, Madrid
POI 3	
id	poi3
provider	provider1
name	Royal Palace
description	Home to the Kings of Spain from Charles III to Alfonso XIII, Madrid’s Royal Palace takes us on a journey. through the history of Spain
location	−3.714302, 40.417974
infoURL	http://www.patrimonionacional.es/real-sitio/palacios/6039
phone	+34 91 454 88 00

Parameter	**Value**
email	info@patrimonionacional.es
address	Calle Bailén , s/n, Madrid
POI 4	
id	poi4
provider	provider1
name	Plaza Mayor
description	This portico lined square is situated at the heart of Hapsburg Madrid, the old part of the city and one of the capitals most charming districts.
location	−3.707398, 40.415364
address	Plaza Mayor, Madrid
POI 5	
id	poi5
provider	provider1
name	El Retiro Park
description	This green oasis in the center of Madrid has 125 hectares and is home to over 15,000 trees. From the botanical point of view, the Park includes some very important gardens.
location	−3.68278, 40.4173
address	Plaza de la Independencia, 7, Madrid

REFERENCES

[1] Chan, P., Dinniwell, R., Haider, M.A., Cho, Y.-B., *et al.* "Inter- and intrafractional tumor and organ movement in patients with cervical cancer undergoing radiotherapy: A cinematic-mri point-of-interest study," *Int. J. of Radiation Oncology*Biology*Physics*, vol. 70, issue 5, pp. 1507–1515, 2008. http://dx.doi.org/10.1016/j.ijrobp.2007.08.055

[2] Erdelj, M., Loscri, V., Natalizio, E., Razafindralambo, T., "Multiple point of interest discovery and coverage with mobile wireless sensors," *Ad Hoc Networks*, vol. 11, issue 8, pp. 2288–2300, 2013. http://dx.doi.org/10.1016/j.adhoc.2013.04.017

[3] Parkinson, B.W., Spilker, J.J., *Global Positioning System: Theory and Applications, Volume One*, Vol. 1, AIAA, 1996. http://dx.doi.org/10.2514/4.866395

[4] Star, J., Estes, J., *Geographic information systems*, prentice-Hall Englewood Cliffs, 1990.

[5] Azuma, R.T., "A Survey of Augmented Reality," *Presence* 6 (4), 1997, pp. 355–385.

[6] Haklay, M., Weber, P., "OpenStreetMap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, issue 4, pp. 12–18, 2008. http://dx.doi.org/10.1109/MPRV.2008.80

[7] Kent, S., "Model Driven Engineering," in *Proc. of the 3rd Int. Conf. on Integrated Formal Methods*, IFM '02, Springer-Verlag, London, UK, pp. 286–298, 2002. http://dx.doi.org/10.1007/3-540-47884-1_16

[8] García-Díaz, V., Fernández-Fernández, H., Palacios-González, E., *et al.*, "Talisman MDE: Mixing MDE principles," *J. of Systems and Software*, vol. 83, issue 7, pp. 1179–1191, 2010. http://dx.doi.org/10.1016/j.jss.2010.01.010

[9] Miller, J., Mukerji, J., Belaunde, M., *et al.*, Mda guide, v1.0.1, Tech. rep., Object Management Group, 2003, [Online]. Available: http://www.omg.org/docs/omg/03-06-01.pdf.

[10] Van Deursen, A., Klint, P., Visser, J., "Domain-specific languages: An annotated Bibliography," *ACM Sigplan Notices*, vol. 35, issue 6, pp. 26–36, 2000.

[11] Asnina, E., Osis, J., „Topological Functioning Model as a CIM-Business Model," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, Hershey - New York, 2011, pp. 40–64. http://dx.doi.org/10.4018/978-1-61692-874-2.ch003

[12] Osis, J., Asnina, E., **„**Derivation of Use Cases from the Topological Computation Independent Business Model." in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, Hershey - New York, 2011, pp. 65–89. http://dx.doi.org/10.4018/978-1-61692-874-2.ch004

[13] TomTom, Tomtom navigator sdk version 3.0 build 193, Tech. rep. 2004. [Online]. Available: http://www.tomtom.com/lib/doc/ttnavsdk3-manual.pdf

[14] Cerf, V. G., "Ascii format for network interchange".

[15] Nolan, D., Lang, D.T., "Javascript object notation," in *XML and Web Technologies for Data Sciences with R*, Springer, pp. 227–253, 2014. http://dx.doi.org/10.1007/978-1-4614-7900-0_7

[16] Wilson, T., Ogc keyhole markup language, 2.2.0, Open GIS Consortium.

[17] Hill, A., MacIntyre, B., Gandy, M., Davidson, B., Rouzati, H., "Kharma: An open kml/html architecture for mobile augmented reality applications," in *9th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR, IEEE, pp. 233–234, 2010. http://dx.doi.org/10.1109/ISMAR.2010.5643583

[18] Lechner, M., Tripp, M., "ARML – an augmented reality standard," Tech. rep., Mobilizy GmbH, 2010.

[19] Madden, L., Professional augmented reality browsers for smartphones: programming for junaio, layar and wikitude, John Wiley & Sons, Inc., 2011.

[20] Mernik, M., Heering, J., Sloane, A.M., "When and how to develop domain-specific languages," *ACM Comput. Surv.*, vol. 37, issue 4, pp. 316–344, 2005. http://doi.acm.org/10.1145/1118890.1118892

[21] Schmid, K., Verlage, M., The economic impact of product line adoption and evolution, *IEEE Software*, vol. 19, issue 4, pp. 50–57, 2002. http://dx.doi.org/10.1109/MS.2002.1020287

[22] Nardi, B.A., *A Small Matter of Programming: Perspectives on End User Computing*, MIT Press, Cambridge, MA, USA, 1993.

[23] Voelter, M., "A catalog of patterns for program generation," in *EuroPloP2003, 8th European Conf. on Pattern Languages of Programs*, 2003.

[24] Efftinge, S., Voelter, M., *oAW xText: A framework for textual dsls, in: Workshop on Modeling Symposium at Eclipse Summit*, vol. 32, 2006.

[25] Eysholdt, M., Behrens, H., "Xtext: implement your language faster than the quick and dirty way," in *Proc. of the ACM int. conf. companion on Object oriented programming systems languages and applications companion*, ACM, pp. 307–309, 2010. http://dx.doi.org/10.1145/1869542.1869625

[26] Des Rivieres, J., Wiegand, J., "Eclipse: A platform for integrating development tools," *IBM Systems Journal*, vol. 43, issue 2, pp. 371–383, 2004. http://dx.doi.org/10.1147/sj.432.0371

[27] Parr, T.J., Quong, R.W., "Antlr: A predicated-ll (k) parser generator," *Software: Practice and Experience*, vol. 25, issue 7, pp. 789–810, 1995. http://dx.doi.org/10.1002/spe.4380250705

[28] Bettini, L., *Implementing Domain-Specific Languages with Xtext and Xtend*, Packt Publishing Ltd., 2013.

[29] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E., *EMF: Eclipse Modeling Framework 2.0*, Addison-Wesley Professional, 2009.

[30] Gronback, R.C., *Eclipse Modeling Project: A Domain-specific Language Toolkit: A Domain-Specific Language (DSL) Toolkit*, 1st ed., Addison-Wesley Educational Publishers Inc., 2009.

[31] Karsai, G., Krahn, H., Pinkernell, C., *et al*., "Design Guidelines for Domain Specific Languages," in *9th OOPSLA Workshop on Domain-Specific Modeling*, 2009. [Online]. Available: http://www.dsmforum.org/events/DSM09/Papers/Karsai.pdf

[32] Meyer, B., *Eiffel: The Language*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[33] Groenniger, H., Krahn, H., Rumpe, B., Schindler, M., Voelkel, S., *Textbased modeling*, in: In: 4th InternationalWorkshop on Software Language Engineering, 2007.

[34] Wile, D., "Lessons learned from real dsl experiments," *Sci. Comput. Program*. vol. 51, issue 3, pp. 265–290, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2003.12.006

[35] Scowen, R., Wichmann, B.A., "The definition of comments in programming languages," *Software: Practice and Experience*, vol. 4, issue 2, pp. 181–188, 1974. http://dx.doi.org/10.1002/spe.4380040211

[36] Wong, S., Cai, Y., Kim, M., Dalton, M., "Detecting software modularity violations," in *Proc. of the 33rd Int. Conference on Software Engineering*, ACM, 2011, pp. 411–420. http://dx.doi.org/10.1145/1985793.1985850

[37] Asnina, E., Osis, J., "Computation Independent Models: Bridging Problem and Solution Domains," in J. Osis, O. Nikiforova (Eds.). *Model-Driven Architecture and Modeling Theory-Driven Development: ENASE 2010*, 2ndMDA&MTDD Whs., SciTePress, Portugal, 2010, pp. 23–32.

[38] Osis, J., Asnina, E., "Is Modeling a Treatment for the Weakness of Software Engineering?" in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, Hershey - New York, 2011, pp. 1–14. http://dx.doi.org/10.4018/978-1-61692-874-2.ch001

[39] Donins, U., Osis, J., Slihte, A., Asnina, E. and Gulbis, B., "Towards the Refinement of Topological Class Diagram as a Platform Independent Model," in J. Osis, O. Nikiforova (eds.). *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2011*, 3rd Whs. MDA&MDSD, SciTePress, Portugal, 2011, pp. 79–88.

[40] Osis, J., Asnina, E., Grave, A., "Formal Computation Independent Model of the Problem Domain within the MDA," *Information Systems and Formal Models, Proc. of the 10th Int. Conf*., ISIM'07, Silesian University in Opava, Czech Republic, 2007, pp. 47–54.

[41] Osis, J., Asnina, E., Grave, A., "MDA Oriented Computation Independent Modeling of the Problem Domain," *Proc. of the 2nd Int. Conf. on Evaluation of Novel Approaches to Software Engineering*, ENASE 2007, Barcelona, Spain, 2007, pp. 66–71.

[42] Osis, J., Asnina, E., "A Business Model to Make Software Development Less Intuitive," *Proc. of the 2008 Int. Conf. on Innovation in Software Engineering*, Vienna, Austria. IEEE Computer Society CPS, Los Alamitos, USA, 2008, pp. 1240–1246.

[43] Osis, J., Asnina, E., Grave, A., "Formal Problem Domain Modeling within MDA," in *Communications in Computer and Information Science*, CCIS, vol. 22, Software and Data Technologies, Springer-Verlag Berlin Heidelberg, 2008, pp. 387–398.

[44] Osis, J. and Asnina, E., "Topological Modeling for Model-Driven Domain Analysis and Software Development: Functions and Architectures," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. IGI Global, Hershey - New York, 2011, pp. 15–39. http://dx.doi.org/10.4018/978-1-61692-874-2.ch002

Vicente García-Díaz is an Associate Professor at the Computer Science Department of the University of Oviedo. He has a Doctoral Degree in Computer Engineering from the University of Oviedo . His research interests include model-driven engineering, domain-specific languages, technology for learning and entertainment, project risk management, software development processes and practices. He is a Specialist in Prevention of Occupational Risks as well as a Certified Associate in Project Management through the Project Management Institute.
Contact address is Computer Science Department, University of Oviedo Edificio de la Facultad de Ciencias. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, España).
E-mail: garciavicente@uniovi.es

Jordán Pascual Espada is a Research Scientist at the Computer Science Department of the University of Oviedo. He has a Doctoral Degree in Computer Engineering from the University of Oviedo, Bachelor Degree in Computer Engineering and Master Degree in Web. He has published several articles in international journals and conferences; he has participated in several national research projects. His research interests include the Internet of Things, exploration of new applications and associated human computer interaction issues in ubiquitous computing and emerging technologies, particularly mobile and web applications.
Contact address is Computer Science Department, University of Oviedo Edificio de la Facultad de Ciencias. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, España).
E-mail: pascualjordan@uniovi.es

B. Cristina Pelayo García-Bustelo is a Lecturer at the Computer Science Department of the University of Oviedo. She has a Doctoral Degree in Computer Engineering from the University of Oviedo. Her research interests include object-oriented technology, web engineering, eGovernment, modeling software with BPM, DSL and MDA.
Contact address is Computer Science Department, University of Oviedo Edificio de la Facultad de Ciencias. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, España).
E-mail: crispelayo@uniovi.es

Juan Manuel Cueva Lovelle became a Mining Engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). He has a Doctoral Degree from Madrid Polytechnic University, Spain (1990). Since 1985 he has been a Professor in the languages and computer systems area at Oviedo University (Spain), and an ACM and IEEE voting member. His research interests include object-oriented technology, language processors, human-computer interface, web engineering, modeling software with BPM, DSL and MDA.
Contact address is Computer Science Department, University of Oviedo Edificio de la Facultad de Ciencias. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, España).
E-mail: cueva@uniovi.es

Janis Osis is a Professor at the Faculty of Computer Science and Information Technology, Riga Technical University, Latvia. He holds *Dr,habil.sc.ing*. degree and is an honorary member of the Latvian Academy of Sciences. The list of publications contains more than 250 titles, including 16 books. During many years his main research interest was topological modeling of complex systems. Recent fields of interests are object-oriented system development, formal methods of software engineering, software development within the framework of MDA by means of topological functioning model support.
Contact address is Department of Applied Computer Science, Riga Technical University, Meža Street 1/3, Riga, LV-1048, Latvia.
Ee-mail: Janis.Osis@rtu.lv