A Framework for Multimedia Data-flow Semantic Annotations

Natalia Díaz Rodríguez, Frank Wickström, Johan Lilius

Turku Centre for Computer Science (TUCS)
Embedded Systems Lab. Department of Information Technologies
Åbo Akademi University, Turku (Finland)
{ndiaz, frwickst, jlilius}@abo.fi

Abstract.

1 Introduction

Modeling complex systems often requires different domain specific languages. Semantic integration makes easier the bridge among synergies.

Defining a language and its generator is considered a difficult task if the aim is building a language for everyone. The task eases considerably if the focus is on one problem domain and domain specific modeling (DSM) can support development and use [10].

2 Domain scenarios to model

...
Another use case of video stream application can be in the bioimaging domain, taking continuous sets of image streams from the microscope for an automatic selection and/or tagging of relevant images. A feedback loop in actions can be modeled with Canals to modify, in real time, the area of interest for which more data is needed. Some data-flow based use cases in NLP can be the automatic annotation and summarization of text. Given large streams of text, topic disambiguation, summaries and semantics can be obtained for faster processing of information.

3 Description of our data intensive context-aware Use case

The diagram in Figure 1 shows how data-stream and event-stream based architectures can be merged to serve such application scenario. The schema uses notation from the Canals language (see next section), and it represents a Canals network which can be mapped to a concrete architecture through its distributors. The network is composed by computation kernels that can be parallelized/scheduled in many-core architectures in order to optimize massive data processing.

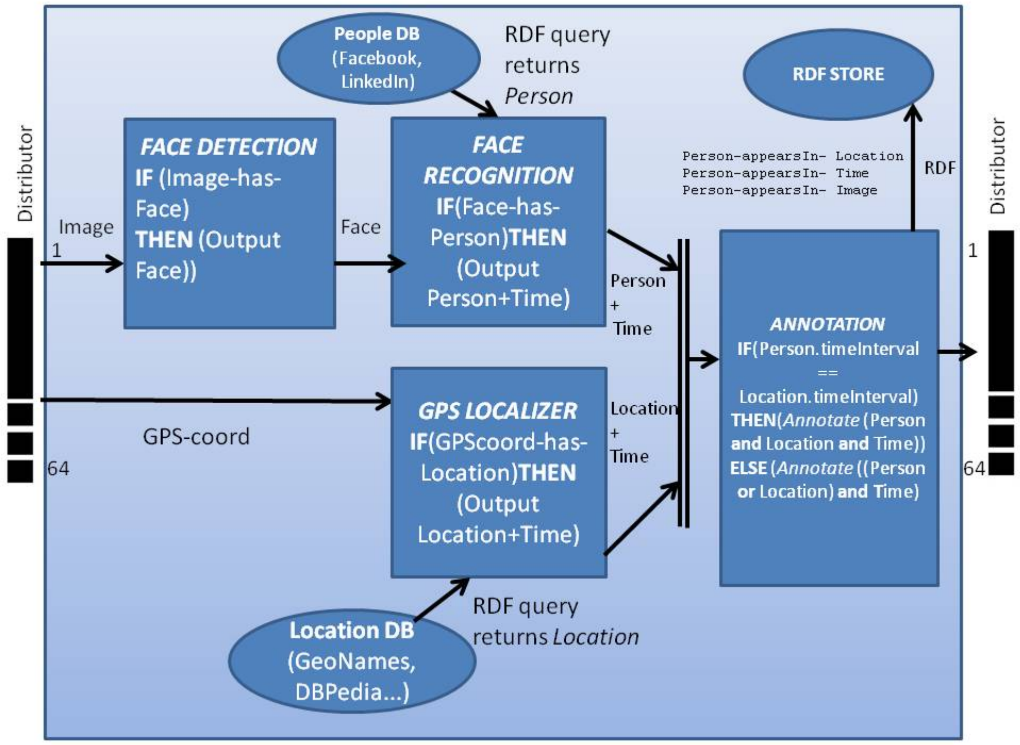

People DB (Facebook, LinkedIn)
RDF query returns *Person*
RDF STORE
Distributor
Image
1
FACE DETECTION IF (Image-has-Face) THEN (Output Face))
Face
FACE RECOGNITION IF(Face-has-Person)THEN (Output Person+Time)
Person-appearsIn- Location
Person-appearsIn- Time
Person-appearsIn- Image
RDF
Distributor
1
Person + Time
ANNOTATION IF(Person.timeInterval == Location.timeInterval) THEN(*Annotate* (Person and Location and Time)) ELSE (*Annotate* ((Person or Location) and Time)
GPS-coord
64
GPS LOCALIZER IF(GPScoord-has-Location)THEN (Output Location+Time)
Location + Time
64
Location DB (GeoNames, DBPedia...)
RDF query returns *Location*

Fig. 1. Data Stream and Event-based face recognition on a video stream.

4 Canals, a data-flow language

Canals [2] is based on the concept of nodes (kernels and networks) and links (channels), which connect nodes together. Computations are performed in the nodes and the links are representing intermediate buffers in which output data from nodes is stored before the data is consumed by the next node. Canals is implemented in Java and generates code in C. Since Canals is a language independent platform, when a Canals program is mapped to an architecture, the platform independent links are mapped to real connectors on the architecture. Connectors are implemented in the Hardware Abstraction Layer (HAL). HAL consists of stateless low-level functions that triggers underlying hardware mechanisms specific for the underlying platform and the functions used by the dispatchers execute kernels, handle synchronization of kernels/processors and triggers memory transfers. A Canals program without a mapping does not depend on specific hardware synchronization concepts but, instead, only describes the intended behavior of a program in a platform independent manner. Summarizing, the main features of Canals are: the ability to describe scheduling algorithms within the language itself, the implicit deserialization of data at network inputs, and an abstraction layer for synchronization between scheduler and computations.

5 A multimedia data-flow semantic annotation model

Our semantic annotation metamodel can be approached in several ways.

1. Taking Canals model as a starting point. The description of the Canals language's semantic extension can be done in two ways:
   a) Extend Canals metamodel with Java code. Since a metamodel must be done for our semantic annotation architecture, in e.g. EMF, the corresponding generated Java classes can be interfaced with Canals.
   b) Make Canals metamodel EMF-compatible (since Canals metamodel was designed ad-hoc, manually) and extend Canals EMF metamodel with our semantic annotation metamodel. Since Canals metamodel is much more complex and large in size, manual adaptations of the generated code might easily become a hassle.
2. Creating an independent semantic annotation metamodel. For this purpose, using UML or OWL modelling languages is a design question. However, a rule engine, production system [1] (i.e. expert system) must be able to become integrated into the system to manage both data-flow and semantic annotation algorithms. Besides, a reasoner will validate the metamodel with, e.g., rules (business rules, etc.).

Taking a bottom-up approach, and focusing on context-aware Smart Space applications, we decide to take approach 2, creating first an independent metamodel in EMF for our semantic framework and providing later an interface with Canals as in option 1.b). This approach leaves flexibility to the semantic annotation metamodel, which could also be run with e.g., another data-flow or parallel language that adapts to certain domain applications, keeping independency in this case of Canals and the semantic annotation framework. The idea is to start modeling a concrete application model for validation of the semantic metamodel, and then, a schedule could be written in Canals for parallelizing and scheduling a data intensive application.

In our use case for multiple video tagging, different Interpreters, Concepts and one Environment containing all these is represented in Figure 2. Time is an entity that we assumed to be embedded in any GPS input data as well as in any frame header from a video. Therefore, all Interpreters (`FacialDetection, FacialRecognition` and `GPSLocator`) can perform a Time annotation in parallel to their specific task. Time could be an independent class also though. In cases where Time is not deductible from the direct Observation, an extra Interpreter, e.g., `TimeInterpreter`, could be used for obtaining Time from other available context data.

The metamodel presented in Figure 2 enables algorithms of different domain to be applied to annotate different kinds of data (image, video, audio,

[1] http://en.wikipedia.org/wiki/Production_system

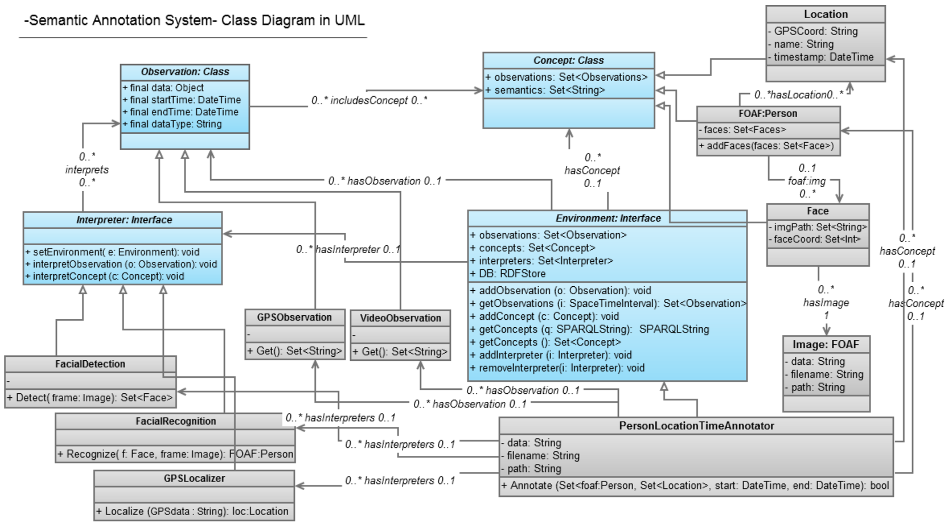

-Semantic Annotation System- Class Diagram in UML
Observation: Class
+ final data: Object
+ final startTime: DateTime
+ final endTime: DateTime
+ final dataType: String
Concept: Class
+ observations: Set<Observations>
+ semantics: Set<String>
Location
- GPSCoord: String
- name: String
- timestamp: DateTime
FOAF:Person
- faces: Set<Faces>
+ addFaces(faces: Set<Face>)
Face
- imgPath: Set<String>
- faceCoord: Set<Int>
Image: FOAF
- data: String
- filename: String
- path: String
Interpreter: Interface
+ setEnvironment(e: Environment): void
+ interpretObservation (o: Observation): void
+ interpretConcept (c: Concept): void
Environment: Interface
+ observations: Set<Observation>
+ concepts: Set<Concept>
+ interpreters: Set<Interpreter>
+ DB: RDFStore
+ addObservation (o: Observation): void
+ getObservations (i: SpaceTimeInterval): Set<Observation>
+ addConcept (c: Concept): void
+ getConcepts (q: SPARQLString): SPARQLString
+ getConcepts (): Set<Concept>
+ addInterpreter (i: Interpreter): void
+ removeInterpreter(i: Interpreter): void
GPSObservation
-
+ Get(): Set<String>
VideoObservation
-
+ Get(): Set<String>
FacialDetection
-
+ Detect(frame: Image): Set<Face>
FacialRecognition
+ Recognize(f: Face, frame: Image): FOAF:Person
GPSLocalizer
+ Localize (GPSdata : String): loc:Location
PersonLocationTimeAnnotator
- data: String
- filename: String
- path: String
+ Annotate (Set<foaf:Person, Set<Location>, start: DateTime, end: DateTime): bool
0..* includesConcept 0..*
0..* interprets 0..*
0..* hasObservation 0..1
0..* hasConcept 0..1
0..* hasInterpreter 0..1
0..*hasLocation0..*
0..1 foaf:img 0..*
0..* hasImage 1
0..* hasConcept 0..1
0..* hasConcept 0..1
0..* hasObservation 0..1
0..* hasObservation 0..1
0..* hasInterpreters 0..1
0..* hasInterpreters 0..1
0..* hasInterpreters 0..1

Fig. 2. Multimedia Semantic Annotation Metamodel and Model in UML.

GPS, accelerometer or sensor data, etc.). The execution of the different annotation algorithms is encapsulated in the `Interpreter` and `Environment` interfaces, and these could be scheduled and parallelized within Canals; i.e. the methods `addConcept` and `addObservation` will realize all data intensive annotation procedures. The first one adds observations to a certain concept being annotated, and the second, can create further grained Observations from given Observations. The interface `Environment` could be one of the only API accessible from Canals code.

6 System Architecture

When designing the architecture for a framework with the features described in the previous section, we believe that DSL can ease the task, while ontologies can better formalize the semantics and validity of our model.

6.1 Domain Specific Languages

Description Logics based ontology languages are getting more accepted as a means for describing concept structures and constraints, e.g., OWL provides a precise description of the underlying concepts as well as a good bases for reasoning over models in certain domain [12]. One goal of DSM is to reduce development time. However, since a DSL is much less expressive than e.g. an UML diagram, model elements can be annotated with OWL text and global conditions can be stated using the Manchester OWL Syntax style. After having the whole domain model and its constraints in abstract syntax, the idea, e.g. in [12], is to project it to a complete ontology for querying and reasoning.

As metamodeling is a well known way to describe languages [1], and modeling languages such as DSLs must be defined formally to be supported by tools (to be able to generate code from them) [6], our objective is to precisely define the DSLs metamodels and find main connectors for their integration. This allows to define domain models and constraints for model elements simultaneously.

6.2 Tools for model design and validation

A model transformation tool [7] that takes as input a UML object model and its class model and produces an ontology, that can be processed by an OWL2 reasoner, can reveal if the object model elements conform to their class model or not.

Two well known metamodeling languages, that target the definition of DSLs, are MOF or UML infrastructures, often used to define the UML or also to define new modeling languages. Other popular metamodeling languages, targeting the definition of DSLs, are Ecore, used in the Eclipse Modeling Framework (EMF) [2], or KM3.

EMFText OWL2 Manchester Editor, is an Eclipse-based, open-source graphical ontology editor with Pellet integration.

6.3 An Ontology for data-flow and context-aware semantic annotations

Ontologies (represented e.g. by OWL or RDFS languages) have the advantages of supporting other (domain specific) languages by enabling validation or automated consistency checking and having a better support for reasoning than MOF (Meta Object Facility)[3] based languages [3].

For the ontology searching task, with the aim of reusing existing work, there are several tools that help us finding present ontologies:

– OMV: Ontology Metadata Vocabulary [4].
– Oyster.
– Swoogle, Semantic Web Search.
– Watson.
– Scarlet - Semantic relations between concepts play an important role in several Semantic Web tasks, such as search, ontology matching, ontology enrichment. Scarlet discovers such relations by exploring the entire Semantic Web as a source of background knowledge. Relying on the functionality of Semantic Web gateways such as Watson or Swoogle, Scarlet automatically finds and combines knowledge provided by multiple online ontologies.

[2] http://www.eclipse.org/modeling/emf/
[3] Meta Object Facility, an OMG language for specifying metamodels
[4] http://omv.ontoware.org/

– Other Ontology Registries.

When a candidate set of ontologies is found, criteria for assessing them needs to be identified.

In [4] a formal context model compounded by four independent ontologies (users, devices, environment and services) is presented for describing intelligent environments and enabling interoperability with external ontologies.

Some interesting aspects modeled in [4] are locations in time, and devices for services and interfaces. This ontology can be reused for the management of embedded resources in the AMEBA project.

The state of the art in available context-aware ontologies is well summarized in [11] as Figure 6.3 shows.

Subdomain \ **Ontology**	CC/PP	Cobra-ONT	CoDAMoS	Delivery Context	SOUPA
Device	X	X	X	X	
Environment		X	X	X	
Interface					
Location		X	X	X	X
Network				X	
Provider					
Role		X	X		
Service			X		
Source					
Time		X	X		X
User	X	X	X		X

Table 1. Subdomain addressed by available context ontologies [11]

We propose an architecture that focuses on observations of certain types, for a context environment to more efficiently process the annotations to be made. Some of the subdomain needed for our domain specific applications are content extraction, online social networks and support for human activity representation.

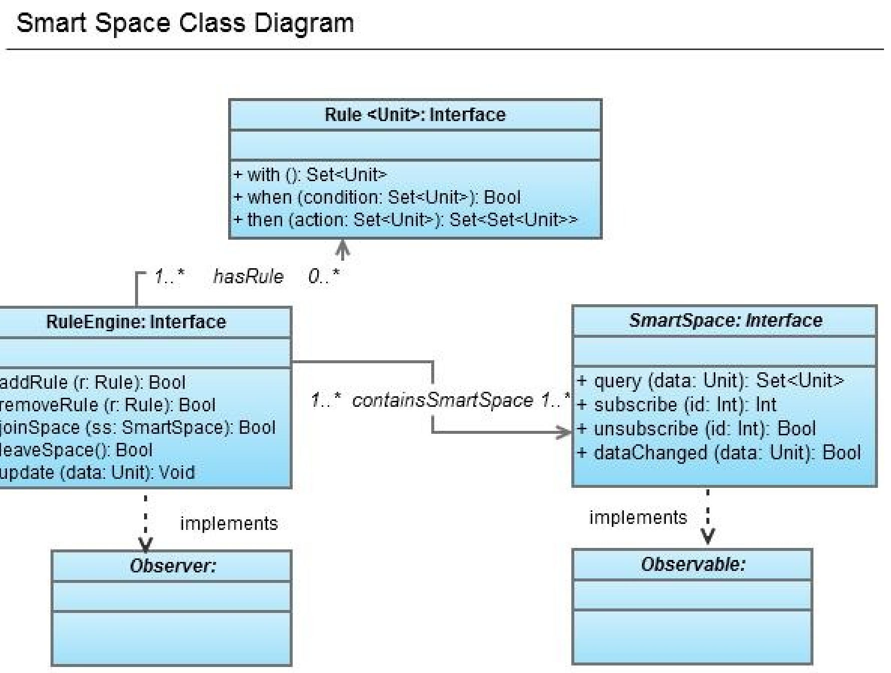
Next, a context ontology for reasoning with annotated metadata is created. Our focus goes into a location-aware multimedia content. This ontology will allow us having, e.g. a hierarchy of location sources, which will allow reasoning and inferring, for example, of the current position of a person. The provenance of different Location tags could come from FourSquareLocation, WalkbaseLocation,CalendarLocation, GPSLocation, OfficeTrafficLightLocation or FaceLocation (using a FacialRecognition Interpreter and obtaining context from the image).

Some elements modeled on our ontology can be reused from existing ones. These aspect are detailed below.

– `Time`: the OWL Time ontology, carried out by the Semantic Web Best Practices and Deployment Working Group (SWBPD) from W3C can be reused. However, the extension in [11] with “Specified Values: Set of Individuals” (LP-SV-01 [15]) pattern allows representing an enumeration of different individuals, e.g., to model the days of the week.
– `Smart Space`: This class represent an abstraction of a physical space, its semantic repository and its protocols to access it, (including subscription mechanisms). This class can be connected with the general Smart Space model designed for a platform independent Smart Space that contains rules for a rule production system. The general model, based on our previous work [9], is sketched in Figure 3. This higher hierarchy of the model specifies a general semantic programming platform whose three main interfaces provide flexibility of 1. database (or RDF store), 2. programming language and 3. rule engine. This approach, therefore, does not only ease to program rules in semantic environments, but also through relational databases, noSQL, quads or others. Analogically, it offers to the programmer freedom of choice in reasoning algorithms and rule engines.
– `Activity`: Human activity recognition based on context-aware ontology reasoning is built on top of our context ontology. Different inferring algorithms can be applied to infer the activity a human is realizing right now.
– `User profile preferences`: CC/PP (Composite Capabilities/Preference Profiles) W3C structure and vocabularies. A CC/PP profile is a description of device capabilities and user preferences [5]. On the other side, a W3C Delivery Context Ontology and a glossary of terms for device independence exists but its maintenance was discontinued.
– `Online social communities`: The SIOC (Semantically-Interlinked Online Communities) [6] Ontology describes information from online communities (e.g., message boards, wikis, weblogs, etc.) on the Semantic Web. Foursquare check-ins can be managed by using SIOC. It is specially of our interest its view of actions (events) in which a user of an online service acts upon one or several objects of that service.
– `Content extraction`: Image Annotation W3C [7] ontology for semantic image annotation and retrieval can be reused.
– `Interfaces, devices and services`: The mIO! context ontology is a network ontology that represents the user context to be able to process and use it to configure, discover, execute and enhance different services that the user may be interested in [11].

There exist specifications for constructing ontology networks [8] when different ontologies can be combined. Following a methodology to guide ontology development is shown useful [11], as well as reusing knowledge resources and attend to good practices in the ontology development. However, reusing con-

[5] http://www.w3.org/TR/CCPP-struct-vocab/
[6] http://rdfs.org/sioc/spec
[7] http://www.w3.org/2005/Incubator/mmsem/XGR-image-annotation/



Smart Space Class Diagram
Rule <Unit>: Interface
+ with (): Set<Unit>
+ when (condition: Set<Unit>): Bool
+ then (action: Set<Unit>): Set<Set<Unit>>
1..* hasRule 0..*
RuleEngine: Interface
+ addRule (r: Rule): Bool
+ removeRule (r: Rule): Bool
+ joinSpace (ss: SmartSpace): Bool
+ leaveSpace(): Bool
+ update (data: Unit): Void
1..* containsSmartSpace 1..*
SmartSpace: Interface
+ query (data: Unit): Set<Unit>
+ subscribe (id: Int): Int
+ unsubscribe (id: Int): Bool
+ dataChanged (data: Unit): Bool
implements
Observer:
implements
Observable:

Fig. 3. General Smart Space Class Diagram.

text ontologies within a new ontology development becomes difficult due to the different purposes and requirements for which the ontologies are designed.

7 Related work

Integration of different DSLs and ontologies using metamodel combination can be seen in [12]. It shows the merge can be loss-free, containing all information from BDSL (DSL for describing network devices) and FODA (feature description DSL). MetaEdit+ is a tool that integrates modeling and metamodeling approaches for DSLs [10].

In [5], metamodels are represented and validated using OWL2 and SWRL. Ordered facts and the closed-world assumptions, available in UML, are represented in OWL2 DL. UML class models and object models are mapped into OWL2 for an automatic validation process with an OWL2 reasoner.

The Meta Object Facility (MOF) [8] has proven itself as a valuable and powerful foundation for a family of modeling languages (such as UML, ODM, CWM, etc). However, MOF 2 suffers from the same structural rigidity as many object-oriented programming systems, lacking the ability to classify objects by multiple

[8] http://www.omg.org/technology/documents/modeling_spec_catalog.htm

metaclasses, the inability to dynamically reclassify objects without interrupting the object lifecycle or altering the object's identity and a too constrained view on generalization and properties. MOF Support for Semantic Structures (SMOF) modifies MOF 2 to support dynamically mutable multiple classifications of elements and to declare the circumstances under which such multiple classifications are allowed, required, and prohibited.

The TwoUse Toolkit is a toolkit that provides features that support the model driven development and the ontology engineering. OWLizer in TwoUse Toolkit can be used for the generation of an ontology from an annotated Ecore model.

8 Conclusions and Future work

Acknowledgment

The research work presented in this paper is funded by the Academy of Finland through the AMEBA project.

References

1. J. Bezivin, C. D. G. No, J. B. Nantes, O. Gerbé, and H. Montréal. Towards a precise definition of the omg/mda framework, 2001.
2. A. Dahlin, J. Ersfolk, G. Yang, H. Habli, and J. Lilius. The Canals Language and its Compiler. *Language*, 2009.
3. H.-J. Happel and S. Seedorf. Applications of ontologies in software engineering. In *International Workshop on Semantic Web Enabled Software Engineering (SWESE'06)*, Athens, USA, November 2006.
4. R. Hervás, J. Bravo, and J. Fontecha. A context model based on ontological languages: a proposal for information visualization. 16(12):1539–1555, jun 2010. `http://www.jucs.org/jucs_16_12/a_context_model_based`.
5. S. Höglund, A. H. Khan, Y. Liu, and I. Porres. Representing and validating metamodels using owl 2 and swrl. In A. Aplinskas, H. Pranevicius, and T. Nakatani, editors, *PROCEEDINGS OF THE 9TH JOINT CONFERENCE ON KNOWLEDGE-BASED SOFTWARE ENGINEERING (JCKBSE'10)*, pages 133–144. Kaunas University of Technology, 2010.
6. S. Kelly and J. Tolvanen. *Domain Specific Modelling*, pages 3–4. John Wiley & Sons, 2007.
7. A. H. Khan, E. Suenson, and I. Porres. Representation and conformance of uml models containing ordered properties using owl2. In E. D. Valle, I. Horrocks, and A. Bozzon, editors, *OrdRing2011 First International Workshop on Ordering and Reasoning*, volume 2011 of *CEUR-WS Proceedings.*, pages 1–12. 10th International Semantic Web Conference (ISWC 2011), 2011.
8. E. M.-P. B. V.-T. Z. Y. G. A. d. C. A. G. M. F.-L. A. G.-P. M. E. M. S. M. C. Suárez-Figueroa, K. Dellschaft. Neon deliverable d5.4.1. neon methodology for building contextualized ontology networks. neon project. http://www.neon-project.org. 2008.

9. M. M. Saleemi, N. Díaz, J. Lilius, and I. Porres. A Framework for Context-aware Applications for Smart Spaces. In *Proceedings of ruSMART 2011: The 4th Conference on Smart Spaces*, 2011.
10. J.-p. Tolvanen and S. Kelly. MetaEdit + : Defining and Using Integrated Domain-Specific Modeling Languages. *October*, pages 819–820, 2009.
11. M. P. Villalon, M. C. Suárez-Figueroa, R. García-Castro, and A. Gómez-Pérez. A context ontology for mobile environments. In *Proceedings of Workshop on Context, Information and Ontologies - CIAO 2010 Co-located with EKAW 2010*, volume 626, Alemania, October 2010. CEUR-WS.
12. T. Walter and J. Ebert. Combining DSLs and ontologies using metamodel integration. In *Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages*, DSL '09, pages 148–169, Berlin, Heidelberg, 2009. Springer-Verlag.