Programming Biomedical Smart Space applications with *BioImageXD* and *PythonRules*

Natalia Díaz Rodríguez
Turku Centre for Computer Science (TUCS)
Åbo Akademi University
ndiaz@abo.fi

Pasi Kankaanpää
Department of Biochemistry and Food Chemistry
University of Turku (Finland)
pkanka@utu.fi

M. Mohsin Saleemi
Turku Centre for Computer Science (TUCS)
Åbo Akademi University
msaleemi@abo.fi

Johan Lilius
Turku Centre for Computer Science (TUCS)
Åbo Akademi University
jolilius@abo.fi

Iván Porres
Department of Information Technologies
Åbo Akademi University
iporres@abo.fi

ABSTRACT

In biomedical sciences new and increasingly complex technologies are constantly being introduced, and to facilitate new scientific discoveries they need to be able to adapt to current demands and provide seamless functionality. However, the interoperability of devices, tools and data is not always simple, and is often a bottleneck in bioimaging. Tools that ease the integration of diverse technologies could amplify functionality and performance considerably. *BioImageXD* software is presented as a use case. We propose a semantic-based approach and *PythonRules* to increase programming productivity in heterogeneous spaces through adaptable and scalable ontology-based development of *Smart Space* applications.

Categories and Subject Descriptors

D.2.m [**Miscellaneous**]: Software Engineering—*Rapid prototyping, Reusable software*

General Terms

Languages, Management, Design, Human Factors

Keywords

BioImaging, Ubiquitous Computing, Smart Space, Ontology, Interoperability, Application Development, Context-Aware

1. INTRODUCTION

Today, biomedical sciences are highly dependent on advanced technologies and computer-controlled systems and analyses, requiring more interaction with each other. New devices are constantly being introduced, often involving different information formats or coming from diverse sources. Lack of interoperability between systems and devices easily becomes a problem, resulting in them not being used efficiently. In order to have devices that interoperate seamlessly, their respective data and functionality must be easily integrated and accessible. The idea of ubiquitous space was proposed as an ideal world where humans and surrounding devices interact unobtrusively. We propose to bring this concept into the realm of biomedical sciences through applying it in bioimaging and *BioImageXD* , thus making the valuable research conducted more functional. **BioImageXD** (BXD) is a free, open source software for visualizing, processing and analyzing multidimensional bioimaging data[1] tailored for advanced fluorescence-based light microscopy.

2. ONTOLOGY BASED APPLICATION DEVELOPMENT IN SMART SPACES

One way to make the work of the researcher easier is having a central data store accessible by all devices and systems present in his everyday life, allowing in this way, integration and sharing of information for the interoperability of different systems. The concept of *Smart Space* can be applied, with this in mind, to promote an easier and more efficient creation of new functionality and to encourage knowledge discovery. Within a ubiquitous space, the enabler technology infrastructure has to be examined. Secondly, from the programmer point of view, the user experience has to be considered i.e., how the *Smart Space* technology can serve the user in daily activities, without much interaction. As a solution for the underlying infrastructure, the idea of *Smart Space* is presented as a physical space that encapsulates its information, allowing devices to join and leave the space, and enabling interoperability in cross-domain scenarios.

Our programming interoperability solution for rapid application development [3, 4] in *Smart Spaces* is based on the open source *Smart-M3* architecture [2]. The platform supports service composition in ubiquitous computing environments through information interoperability[5]. **Smart-M3** (Multi part, Multi device, Multi vendor) architecture [2] provides a particular implementation of *Smart Space* where the

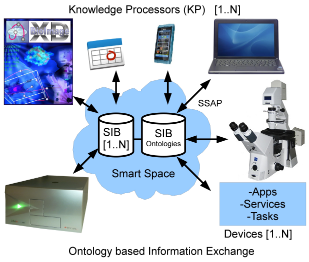

Figure 1: A bioimaging *Smart Space* with *Smart-M3* architecture provides information interoperability between phone, PC, microscope, microplate reader and applications such as *BioImageXD* and calendar.

central repository is the Semantic Information Broker (SIB). Information is processed by entities called Knowledge Processors (KPs) which interact with the space through RDF and the SSAP protocol (Fig.1).

We use an ontology-driven development approach for mapping ontologies to object oriented (OO) programming [2, 3]. Our system consists of two parts. The first one, the ontology library, generates a Python API from an OWL ontology, producing classes, methods and declarations which can be used by the KP developer to access data in the SIB. The second part is the middleware which abstracts the communication with the SIB. Its aim is the handling of RDF Triples with the generated API, providing functionality for synchronous and asynchronous queries, enabling application developers to program new KPs without worrying about the SIB interface. Given this functionality, there is a need for designing a rule syntax that allows users - with basic programming skills – to easily define rules to model smart applications. With this idea in mind, our aim is to minimize the need for learning OWL, RDF or query languages. The main aim of *PythonRules* is to encapsulate, to the programmer of the *Smart Space*, the communication interface. The programmed inference rules evolve by KPs taking part in the application; they can define, e.g., how to infer the user's activity using the active context information from multiple KPs (e.g., how to handle emergency calls while being at a meeting).

A **PythonRule** is defined with a 3-clauses pattern such as **With()//When()>>Then()**. The **With()** Clause represents *assertions* or *declarations* of individuals; the **When()** Clause represents *conditions* or *events* that must hold before the triggering and the **Then()** Clause represents *actions* to trigger or inferred *conclusions*. *PythonRules* enable Python to include logic statements representing inference. The abstraction of the *PythonRules* comes from translating Python statements to *RDF* queries through the SSAP protocol, who is also responsible of the execution of the rules through a subscription mechanism.

Having the integration of ontology-based logic programming with the OO paradigm as main aim, our development framework consist of 1. *Smart-M3 Ontology to Python API Generator* [2]. 2. Middleware to abstracts the communication with SIB. 3. *PythonRules for Smart Space programming* to ease interaction with the repository and handling of namespaces providing a higher abstraction for fast specification of the space's behavior. When the programmer does not deal with RDF directly but mainly with Python statements, the translation of problems into programs becomes easier. *PythonRules* could be used to program, according to the researcher's calendar application, when to start running time-consuming or complex experiments, remotely. An alternative to the classical way of processing data in batch mode is using the ontology generated Python classes, which trades off the shortness of code for the command readability, aiding also at data sharing and process automation. Below, the rule is defined in this way:

```
with = With([labWorkerCalendar, meetingEvent,
    bxdRun, meanFilter])
when = When(meetingEvent.getProperty("Title")
    == "Lab Weekly Meeting")
then = Then([bxdRun.new(BXDBatchAnalysis),
bxdRun.setProperty(ProcessID = 86,
 bxdRun.setProperty(ProcessDate = today),
 bxdRun.setProperty(FileIN ="./sample.lsm"),
 bxdRun.setProperty(FileOUT ="./resultsFile"),
 bxdRun.setProperty(Timepoints= "1"),
 bxdRun.setProperty(Channels= "1",
 bxdRun.setObject("AppliesFilter", meanFilter),
 bxdRun.runCommandLine(twoDaysBeforeMeeting)])
rule = PythonRule(with, when, then)
```

3. DISCUSSION

Diverse works show the usability and power that ontologies can bring to data acquisition and interpretation, e.g. to facilitate registration and analysis of images. Rule specification is often done through SWRL or SPARQL, while we introduce comparable functionality by ”abstracting away” the technical details of RDF and query languages. Fast prototyping of mashup applications becomes easier. Our proposal addresses the challenge of context-aware ubiquitous computing by using automated ontology code generation facilitating OO programming of *Smart Spaces*. The toolchain applied to bioimaging domain showed how ontology-based development may require extra preprocessing, that ends paying off by assisting in automating processes. *PythonRules* provides software power in a more comprehensive interface, in which learning OWL, RDF or query languages is not needed for taking advantage of the Semantic Web's benefits.

Acknowledgments

This research is funded through the DIEM project by Tekes. The authors thank Lassi Paavolainen, University of Jyväskylä, Finland, for his help with *BioImageXD*.

4. REFERENCES

[1] BioImageXD, 2006. www.bioimagexd.net.

[2] Smart-M3, http://sourceforge.net/projects/smart-m3/.

[3] A. Kaustell, M. M. Saleemi, T. Rosqvist, J. Jokiniemi, J. Lilius, and I. Porres. Framework for Smart Space Application Development. In *IWSI 2011*.

[4] M. M. Saleemi, N. Díaz, J. Lilius, and I. Porres. A Framework for Context-aware Applications for Smart Spaces. In *ruSMART*, 2011.

[5] M. M. Saleemi, N. Díaz, E. Suenson, J. Lilius, and I. Porres. *Semantic Interoperability: Issues, Solutions, Challenges.*, chapter Ontology driven Smart Space Application Development. To appear, 2011.