Tool Support for Designing CML Based Context Models in Pervasive Computing

Johnson Fong[1, 2], Jadwiga Indulska[1, 2] and Ricky Robinson[1, 2]
[1] The University of Queensland
School of Information Technology and Electrical Engineering
[2] National ICT Australia (NICTA)
Queensland Research Laboratory, Australia
{jfong, jaga}@itee.uq.edu.au, ricky.robinson@nicta.com.au

ABSTRACT
Development of context-aware applications is very complex as context information is gathered from a variety of sources that are error prone and differ in the quality of information they provide. The current software engineering approach to building context-aware applications employs appropriate context modelling and reasoning techniques to facilitate the application development process. However, the design of context models is highly challenging, as they need to capture (i) a variety of context information required by the applications (such as location, activities and devices, etc.), (ii) the relationships and dependencies among them, and (iii) their ambiguity, timeliness, and (iv) required Quality of Information. This paper describes a semi-automatic, forward engineering context modelling tool that provides design support for developing context models designed using the Context Modelling Language (CML). It promotes reuse of existing context information and automatically generates code for mapping context models into their runtime representations. The goal is to enable an error resistant context model design process and significantly reduce applications development effort and time.

Keywords
context model, design tool support, context-aware applications, context management

1. INTRODUCTION
Context-aware applications use contextual information to evaluate whether there is any change to the situation of the user and/or the computing environment that requires them to adapt their behaviour [1]. Much of the early efforts in developing context-aware applications have been adopting an *ad hoc* approach to context gathering and evaluation, without explicit attention to the use of context models and middleware that manages and understands the models. There are several problems with this approach. It makes the building of context-aware applications very burdensome for developers, since they have to directly deal with (i) low-level context sources (e.g., sensors), (ii) pre-processing of raw context data to the level required by the application, (iii) evaluation of context information and user preferences to make decisions on how applications should adapt, (iv) differences in Quality of Information of context information, and (v) providing fault tolerance of sources of context information. An *ad hoc* approach is also not amenable to reuse of context information. Moreover it leads to development of applications that have limited evolvability as context evaluation is hardwired into the logic of the application and any modification to the context or user preference models requires re-programming of the application [9].

For these reasons most of the current software engineering approaches to developing context-aware applications use (1) high level, abstract context models that define types of context information required by particular applications and (2) context management systems (CMS) that belong to the middleware level and are able to gather context information, reason about context and evaluate user preferences. This approach significantly reduces the complexity of the applications and the difficulty of their design and implementation, as the burdens of providing context provisioning and management services are moved from the applications to the CMS [1, 2, 3]. In addition, complex issues such as dealing with Quality of Information, conflict resolution, recognising situations that require application adaptations, fault-tolerance, security and user privacy, etc., [1,3,15] are also externalised into the middleware. Context models that model context information used by particular applications also support context reuse (at the time of modelling and also at application run time). All these features of context model based context-aware applications make software engineering of such applications easier and can lead to more evolvable and scalable context-aware applications. The software engineering of context-aware applications can be further assisted by the development of context modelling tools that provide assistance in the design of context models. Unfortunately, the design of context models is inherently challenging and error prone, as they need to capture all context information types required by the applications such as location, activities, computing resources and connectivity, etc. This context information may be imprecise, ambiguous, or not available since it is gathered from a wide variety of context sources (i.e., sensors) that differ in the Quality of Information they produce and failure rates. The context models need to capture this imprecision or ambiguity. Furthermore, the complexity of context models is

UPC2010 Workshop in conjunction with ICPS2010, July 13-15, 2010, Berlin, Germany.

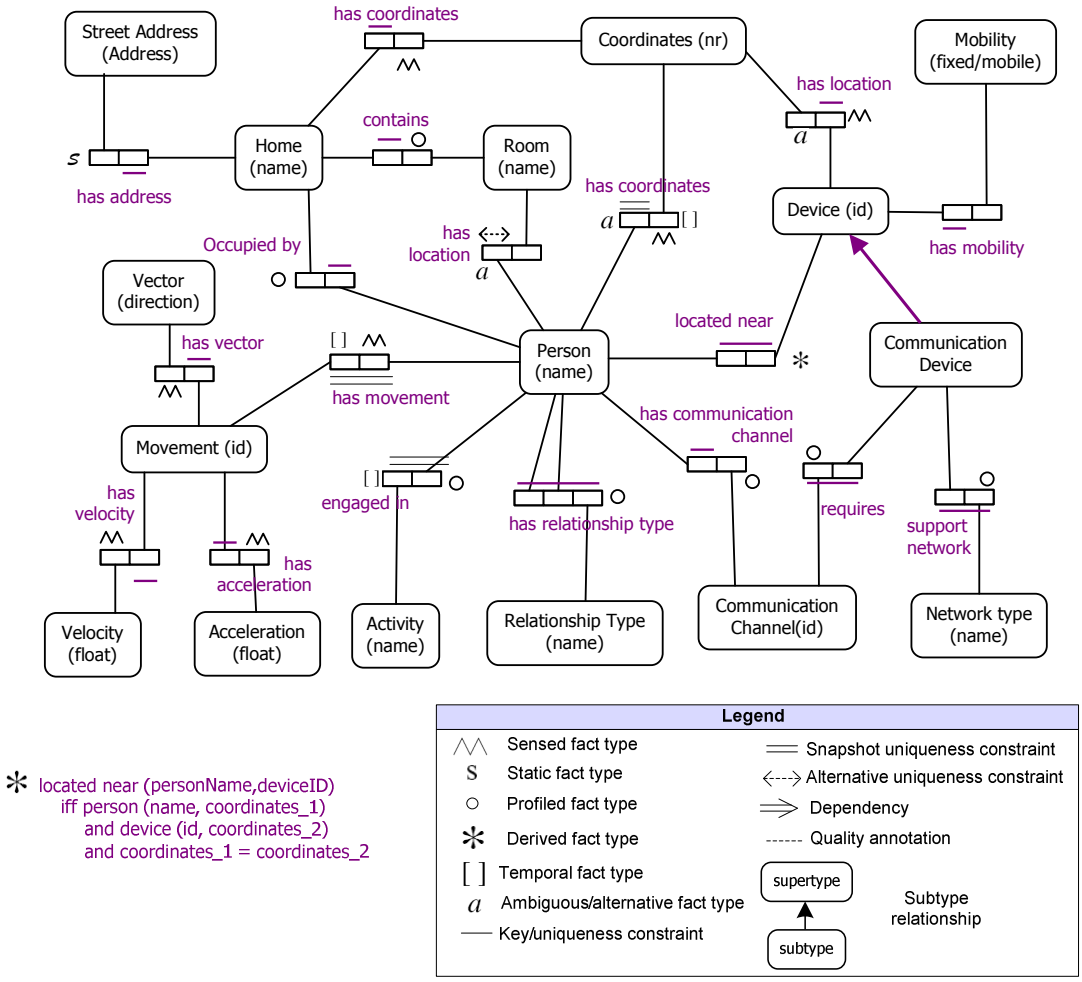

Figure 1. An example CML context model for Context-Aware Virtual Community Applications [5]

exacerbated as applications evolve, because consequent changes to their context models may introduce additional errors, the models may become more complicated and interconnected with context models of other context-aware applications.

There exist a number of high level context modelling approaches, such as ontology based models [1], object-role based models [6], spatial models [16] and rule-based models (e.g., using Unified Modelling Language enriched with Object Constraint Language [17, 18]). In this paper, we describe a context modelling tool that provides design support for developing object-role based context models. The object-role modelling approach that we use to illustrate the functionality of this tool is CML (Context Modelling Language) [3]. The tool supports the following tasks and features: (i) it is capable of constructing, editing, combining and browsing context facts and whole CML context models, (ii) promotes reuse of context information by allowing designers to drag and drop existing context facts from other applications arbitrarily onto their current working model, (iii) it is capable of performing real-time internal model validation to ensure context models can be automatically transformed into their CMS run-time representations according to the recent ORM2 (Object-role Modelling) specifications [4], and finally, (iv) it is a light weight, open-source, completely stand-alone and cross platform application.

The context modelling tool has been inspired by previous work in automation of pervasive computing applications and information modelling, specifically tools that facilitate the development of ORM2 models, as CML is an extension of ORM2. Some of these tools include Neumont ORM Architect (NORMA) [12], Microsoft's Visio for Enterprise Architects[13], Protégé [14] and Rational Rose [21]. These tools are designed for modelling information (some with the use of an ontology, UML and OCL), while the context modelling tool is specifically developed for the sole purpose of building a CML context model graphically and automatically generate code for mapping the validated context model to its runtime representation. The structure of the paper is as follows. Section 2 briefly introduces the CML context modelling approach and shows an example context model defined for a context-aware smart home application. Section 3 explains how CML models can be created and displayed in the context modelling tool, how the tool supports model validation and how the tool automatically generates code to map CML context models to their runtime representations. Finally, Section 4 concludes the paper and outlines future work.

2. CONTEXT MODELLING APPROACH

This section describes the CML context modelling approach. The Context Modelling Language [3], or CML is derived from a conceptual information modelling formalism called ORM (Object Role Modelling) which was developed for modelling information systems and has an intuitive graphical representation. The ORM modelling concept is based on facts, and the modelling of a domain involves identifying fact types i.e., relationships between entities. CML extends the ORM facts to capture additional information that is important for managing context information. The CML extensions include [3]:

- context information type
  - static (S) (e.g., family relationship),
  - profiled (○) (e.g., ownership of a device),
  - sensed (⩚) (e.g., current temperature),
  - derived (✻) (derived from existing context information),
- timeliness ([ ]) to capture histories of context information (e.g., user activity over a period of a week),
- ambiguity/alternative (*a*) that describes ambiguous information (e.g., conflicting location reports gathered from a variety of location sensors), and
- quality (.....) that defines the required QoI of the context information.

An example CML model for a context-aware virtual community application is shown in Figure 1. It is a modification of a context model in [5]. Each soft rectangle in the figure depicts an object type, while each box denotes a role played by an object type within the fact types. For example, the "*occupied by*" fact type contain two roles, one played a "*Person*" object type and the other by the "*Home*" object type. The fact types models are based on ORM. The CML extensions to ORM are summarised in the Legend in Figure 1 and illustrated in the model. For example the model shows that the fact type that the person is located near device is a derived fact type.

Since context-aware applications adapt to changes in situations and not to changes in single sensor values, CML fact types may be combined to formulate higher level abstractions known as *situations*. Situations in this approach are expressed in a modified first order logic. They can be used by triggers that execute when a particular situation arises. Preferences are also evaluated when situations arise to yield one or more execution choices when a decision needs to be made by the context-aware application [6].

3. CONTEXT MODELLING TOOL

The CML object-role based approach described in section 2 supports development of context-aware applications by allowing to model context information types required by the applications. A graphical notation is used in this modelling approach. In this section we describe our context modelling tool aimed at supporting software engineering of context-aware applications by making the design process of application context models easier and more error resistant.

3.1 Functionality

The tool provides means to define graphical representations of context facts as defined in the CML object-role based context modelling approach. It also allows annotating the model with types of context information, Quality of Information, timeliness, ambiguity, and dependencies as defined in the CML model. The tool has the following features:

- Supports context reuse - the designers can create facts or incorporate existing context facts or context models that are already in the repository of context models. The tool supports browsing of the repository and a selection of the models of interest.
- Has an intuitive and user friendly interface for the designers of context models.
- Supports basic validation of CML context models in a semi-automatic fashion.
- Provides forward engineering support for mapping of CML context models to their relational runtime representations used by the context management system.

More detailed descriptions of the user interface, context model validation, mapping to runtime representation, and also the tool implementation are provided in the following subsections.

3.2 User Interface

Figure 2 shows a screen shot of a CML model created using the context modelling tool. The shapes of the object types (as soft rectangles) and connectors in the CML model adopt the latest ORM2 notation specified by Halpin et al [4]. The CML model can be created graphically by selecting the relevant shape from the designer palette located on the left side of the screen, placing it onto the canvas, double-clicking the shape to enter relevant data, and connecting the shape using appropriate connector.

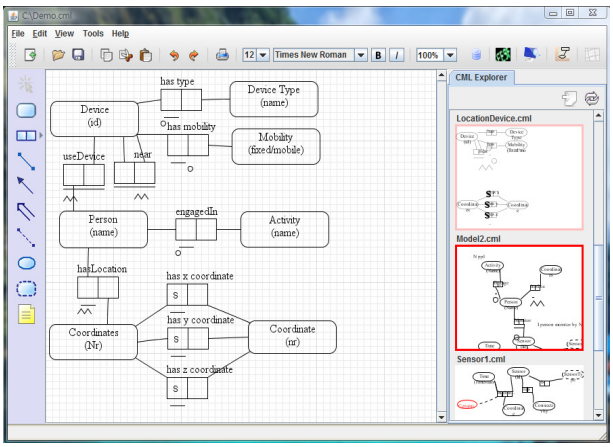

Figure 2. CML model display in context modelling tool

For advanced users, it is also possible to enter object types by using connectors only without the need for continuously selecting between various shapes and connectors (for further explanations and details see [8]).

There are various options on the top right of the screen, which include:

- Automatic mapping to relational schema and SQL scripts (runtime representation of the context model).
- Browsing existing context facts and models using CML explorer as shown in Figure 2. It allows designers to browse through directories for existing context facts, and drag-n-drop facts on to the current working model. The aim is to reuse context facts (or combination of context facts) from other context-aware applications. Designers can choose to hide the side bar to allow more space on the canvas.
- View-port for a bird's eye view of the entire context model when the model becomes too large.
- Titled background for designer to eliminate the background grids that appear on the canvas.

3.3 Model Validation

In order to ensure the integrity and consistency of the context information that is gathered by CMS according to the context models of context-aware applications, the context models have to be correct and valid. The context modelling tool is capable of performing basic correctness checking of the model at design time, to ensure that all the entities and relationships in the context model can be correctly transformed into run-time representations, and eventually be instantiated by the CMS. As illustrated in Figure 3, erroneous shapes are highlighted and when one of the shapes is double clicked, the errors are listed in a new window under the Error Listing section.

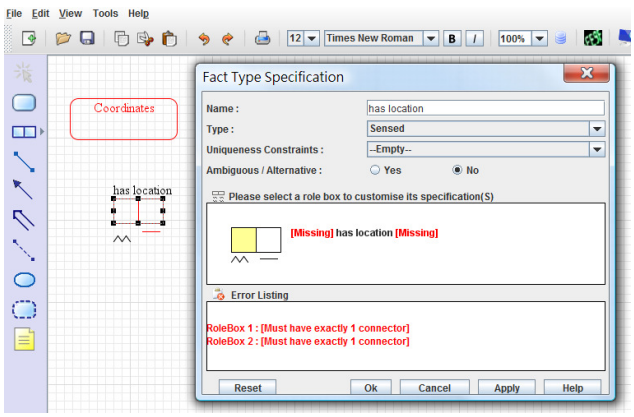

Figure 3. Erroneous shapes and error messages are highlighted

The tool is capable of checking typical errors including omitting constraints (e.g., each object type requires a representation, and each representation requires a data type), and including contradictory constraint combinations (e.g., a subtype cannot be the direct or indirect supertype of its supertype) as shown in Figure 4.

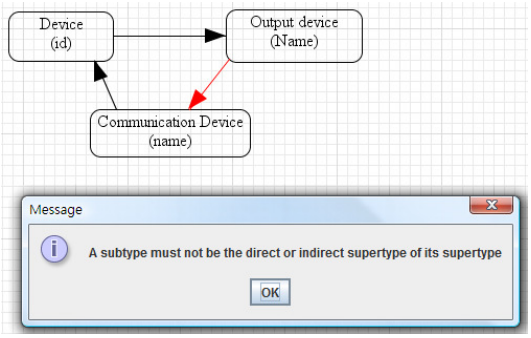

Figure 4. Live error handling

3.4 Code Generation

A context management system is responsible for gathering, evaluating, and provisioning context information according to the context model developed for the context-aware application [10]. Thus, CML context models of particular applications need to be mapped into the relational schemas required for the relational database-style repository of the context management system. Manual transformation of graphical representations of context fact types involves heavy database programming and sound knowledge in context information modelling from designers. It is not uncommon for designers to introduce errors during the transformation process, such as misnaming fact type relations, omitting integrity constraint and other fact specifications that could go undetected until the application encounters errors when manipulating context information. To reduce the database coding and administration effort of the designers in mapping abstract context fact types into their run-time representations, and make the mapping process less error prone, we provide this mapping as one of the functionalities of our tool. The tool is capable of automating the transformation process of graphical CML context models to relational schemas and/or SQL scripts for a specific underlying context management system.

```
Create Table Person
(
     PersonName       varchar(20)   not null,
     CoordinatesNr    smallint      not null,
     HomeName         varchar(20)   not null,
     RoomName         varchar(20)   not null,
     ChannelId        smallint      not null,
     Primary Key    (PersonName),
     Foreign Key (ChannelId) reference
           ChannelRequiresDevice (ChannelId)
);

Create Table Device
(
     DeviceId         smallint      not null,
     CoordinateNr     smallint      not null,
     Mobility         char(1)       not null
           check (Mobility in ('f','m')),
     Primary Key    (DeviceId)
);

Create View PersonLocatedNearDevice
           (PersonName, DeviceId) as
     select A.PersonName, B.DeviceId
     from   Person as A join Device as B
     on A.CoordinatesNr = B.CoordinatesNr;
```

Figure 5. Portions of context model to SQL mapping.

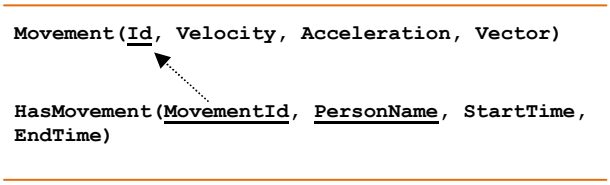

Figure 6. Portions of context model to relational schema mapping.

Figure 5 and 6 illustrate a partial mapping of the CML model (shown in Figure 1) to SQL scripts and a relational scheme, respectively. The translation employs the recent ORM2's relational mapping procedure, Rmap [4], where fact types are lassoed into groups to form a relation, and foreign keys are being introduced to attributes. The main benefits for employing the ORM2 Rmap procedure are (1) to guarantee a redundancy-free relational design, restricting the number of relation to allow effective and efficient database operation, as it is based on the "Optimal Normal Form" (ONF) algorithm, and (2) to maintain referential integrity. The translation build on top of the Rmap procedure by incorporating additional mappings for the context modelling constructs introduced by CML as described in [11].

The graphical CML context models created using the tool are serialized as XML, and stored with the .CML file extension by the context management system. However the automatic code generation is done based on the graphical model. The code generation feature is handled by the class `GenerateSqlScriptActionListener`, which can be found in [7]. Code is generated based on the model that is currently on the canvas at the time the action is being invoked. It is not necessary for designers to have any database programming knowledge to generate the code.

3.5 Tool Implementation

During the early stage of the tool development, a number of implementation technologies have been considered, such as the Eclipse GMF (Graphical Modelling Framework) that is based on the SWT toolkit, and C# plug-in to Microsoft Visual Studio. NET, etc. Eventually, the Java Swing GUI toolkit has been employed for software implementation, aiming to achieve the following characteristics:

- Stand-alone (i.e., not necessary for designers to purchase MS Visual Studio or download Eclipse merely to execute the program).
- Portable/cross-platform (i.e., designers must use MS Windows to run the tool if it was implemented as a plug-in to MS Visual Studio. NET)
- Lightweight (i.e., the tool offers essential functionalities with minimal memory footprint and file size, which can be easily distributed over the web, or even as a web-based Java Applet with all the benefits of Java technology, including robust memory management and security)
- Usable and Efficient (i.e., the tool is designed to have high usability to support efficient and easy production of context models)
- Open-source (i.e., the distribution is made available under the BSD license, and downloadable at sourceforge.net via [7])

4. CONCLUSION AND FUTURE WORK

This paper provided an overview of a context modelling tool that support designers in the development of CML context models for context-aware applications. The tool provides engineering design support for modelling context information, reusing existing context facts and automating the application design process. It is capable of constructing, editing, combining and browsing of context models, supports forward engineering of automatic mapping of context models into their runtime relational representation, and therefore, significantly reduces the database coding and administration effort, and makes the mapping process less error prone.

Efficiency and accuracy of the code generation process can be improved by utilising techniques that provide formal representation for describing a CML model and transforming the model to CMS specific code. We will investigate the use of XCML meta model proposed by Robinson et al [19] and Rails migrations that describe transformations using Ruby [20]. A formal technique will also be investigated for internal validation of the models before generating code. A Domain Specific Language (DSL) can be considered for defining syntax and rules to enforce integrity constraints and maintain consistency of the model.

Long term plans include support for verbalisation of the context model for non-technical users to interpret and understand the model, similar to the external model validation feature found in NORMA [12]. The model's fact types, constraints, and derivation rules are to be verbalized in language that is easily understood by ordinary users, so as to enable users to modify the models, introduce new facts and situations, and ultimately personalise the behaviour of applications according to their individual needs and preferences. The aim is to improve intelligibility and control of context-aware applications.

5. ACKNOWLEDGMENTS

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program; and the Queensland Government.

6. REFERENCES

[1] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni, "A Survey of Context Modelling and Reasoning Techniques", *Pervasive and Mobile Computing*, Volume 6, 2010

[2] P. Hu, J. Indulska and R. Robinson, "An Autonomic Context Management System for Pervasive Computing", *Proc. of the IEEE International Conference on Pervasive Computing and Communications* (PerCom'08), Hong Kong, March 2008.

[3] K. Henricksen and J. Indulska, "Developing Context-Aware Pervasive Computing Applications: Models and Approach", *Pervasive and Mobile Computing*, Volume 2, 2005.

[4] T. Halpin and T. Morgan, *Information Modelling and Relational Databases*, Morgan Kaufmann, Second Edition, 17 March, 2008

[5] J. Indulska, K. Henricksen, T.Mcfadden and P. Mascaro, “Towards a Common Context Model for Virtual Community Applications”, *Proceedings of 2nd International Conference on Smart Homes and Health Telematics (ICOST),* Volume 14 of Assistive Technology Research Series. IOS Press, pp. 154-161, 2004

[6] J. Indulska and R. Robinson, "Modelling Weiser’s “Sal” Scenario with CML", the *Sixth Workshop on Context Modelling and Reasoning* (CoMoRea'09) affiliated with the IEEE PerCom’09 conference, Galveston, US, March 2009, IEEE Press.

[7] "Context Modelling Tool", Sourceforge.net, http://sourceforge.net/projects/contextmodel/, last accessed 25 Feb. 2010

[8] "CMT Demo", Screencast.com, http://www.screencast.com/t/MTYyYzJh, last accessed 14 Mar. 2010

[9] A. Dey, G. Abowd and D. Salber, “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”, *Human-Computer Interaction*, Vol 16: 2, pp. 97-166, 2001.

[10] T. McFadden, K. Henricksen and J. Indulska, "Automating context-aware application development", *UbiComp Workshop on Advanced Context Modelling*, Reasoning and Management, pp. 90-95, 2004

[11] K. Henricksen, J. Indulska and A. Rakotonirainy, "Generating context management infrastructure from context models", in *4th International Conference on Mobile Data Management (MDM) - Industrial Track,* Melbourne, 2003

[12] M. Curland and T. Halpin, " Model Driven Development with NORMA", in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 07),* IEEE, 2007

[13] T. Halpin, K. Evans, P. Hallock and B. Maclean, *Database Modeling with Microsoft’s Visio for Enterprise Architects,* first edition. Morgan Kaufmann, August 28, 2003

[14] "Protégé", Stanford Center for Biomedical Informatics Research, http://protege.stanford.edu/, last access 17 Aug 2009

[15] R. Wishart, K. Henricksen and J. Indulska, "Context privacy and obfuscation supported by dynamic context source discovery and processing in a context management system". *Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing*, Lecture Notes in Computer Science, Vol. 4611, Springer, pp. 929-940, Hong Kong, July 2007.

[16] C. Becker and D. Nicklas, "Where do spatial context-models end and where do ontologies start? A proposal of a combined approach", *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*, University of Southampton, Nottingham, England, 2004.

[17] L. Daniele, P. Dockhorn Costa and L. Ferreira Pires, "Towards a Rule-Based Approach for Context-Aware Applications", *Proceedings of the 13th EUNICE Open European Summer School 2007 (EUNICE 2007)*, LNCS 4606, July 2007.

[18] J. Gaber. "Spontaneous emergence model for pervasive environments". *IEEE Globecom workshops*, Washington DC, USA, 2007.

[19] R. Robinson, K. Henricksen and J. Indulska, “XCML: A Runtime representation for the Context Modelling Language”, in *Proceedings of the 4th International Workshop on Context Modelling and Reasoning (CoMoRea 2007)*, White Plains, NY. March 2007.

[20] K. Marshall, C. Pytel and J. Yurek, *Pro Active Record: Databases with Ruby and Rails,*1st Edition, Apress, 2007.

[21] "Rational Software", IBM.com, http://www-306.ibm.com/software/rational/, last accessed 14 Mar. 2010