

ULRR

Application development using modeling and dynamical systems analysis

Item Type	Meetings and Proceedings
Authors	O'Neill, Eleanor;McGlinn, Kris;Lewis, David;Bailey, Eoin;Dobson, Simon;McCarthy, Kevin
Citation	CAMS '09 Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009;pp. 18-23
Publisher	Association for Computing Machinery
Download date	2026-05-13 12:50:11
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/2390

Application Development using Modeling and Dynamical Systems Analysis

Eleanor O'Neill, Kris McGlinn,
David Lewis
Knowledge & Data Engineering Group
School of Computer Science and
Statistics
Trinity College Dublin, Ireland
{eleanor.oneill, kris.mcglinn,
dave.lewis} @cs.tcd.ie

Eoin Bailey, Simon Dobson
Systems Research Group
School of Computer Science and
Informatics
UCD Dublin, Ireland
{eoin.bailey,
simon.dobson}@ucd.ie

Kevin McCartney
Centre for Architectural Education
College of Science, Engineering and
Food Science
University College Cork, Ireland
k.mccartney@ucc.ie

ABSTRACT

Research on context aware systems is handicapped by the lack of readily available large scale data sets, as well as by the lack of tools by which researchers can interact effectively with such data sets across a range of scales and granularities. We show how virtual reality combined with a dynamical systems analysis approach can start to address these gaps. Simulation allows for generation of simulated sensor data at runtime and actuations of entities in the virtual world. The ease of sensor deployment and configurations in a simulated environment allows for rapid reconfigurations enabling generation of the required large scale data sets for analysis. Using these data sets, dynamical systems analysis can determine if a given application is functioning in a manner that is deemed to be correct.

Categories and Subject Descriptors

D.2.5 [Software Engineering]: Testing and Debugging – Testing Tools. I.6.6 [Computing Methodologies]: Simulation and Modelling - Simulation Output Analysis

General Terms

Experimentation

Keywords

context-aware systems, dynamical system analysis, simulation

1. Introduction

Context awareness and sensor technology are well suited to a wide range of environment monitoring applications from disaster detection to wireless compliance monitoring. Almost a decade ago, Cerpa et al referred to environment monitoring as a 'motivator' for wireless technology [14], pointing out that these systems can generate data at granularities and quantities not previously feasible. Two ongoing problems for developing these systems are firstly that working with large data sets quickly becomes prohibitive without sophisticated analysis tools. The second hurdle is the availability of deployed sensor networks to generate these large data sets; many wireless sensor technologies

remain costly and time-consuming to deploy.

Challenges such as successful reasoning about spatial relations and spatial logics are key issues that need to be addressed for a context-aware application to be accepted by end users. Sensor rich environments differentiate these applications from existing mobile and distributed systems by enabling them to adapt to end user needs. For the system developer, the problems of heterogeneity and scalability are felt most keenly when designing this adaptive behaviour. A context-aware system needs to operate reliably over the wide variety of situations that may be encountered. The context-aware system discussed for testing purposes in this paper, is a simple temperature sensing system used to control windows in an office space.

For the context-aware system designer, knowing how the smart space and the application interact is invaluable information. In complex context aware systems, the scale and heterogeneity of data generated during testing quickly becomes unmanageable. Developers need support to investigate both the context space (inputs) for an application and its behavioural space (outputs). Dynamical systems theory is an area of applied mathematics that has been used in the analysis of complex non-linear systems [6], such as analysis of supply chains [2]. In order to analyse the contextual and behavioural space of an application using this approach, a significant amount of post-experimental data must be collected including information about window actuator control, ambient room temperature and localised room temperature.

Gaining access to large data sets persists in being a problem for developing analysis tools however simulation offers a way to overcome this hurdle. Simulation has repeatedly been used by researchers for developing and testing prototype systems e.g. Ubiwise [9], Topiary [10] and Pudecas [22]. Simulation offers a significant benefit for early stage testing of context-aware prototype systems because it lowers the barrier to entry and supports rapid testing cycles. 3D rendering engines are particularly good at simulating context around location and spatial relations such as containment, proximity and orientation, key factors identified by Bandini et al [15]. However simulating other environment conditions is not standard in many rendering engines. Additional modelling is required to generate context about conditions related to environment monitoring, such as airflow, this is discussed further in section 3.1.

In this paper we discuss the role of a context generating simulator and a context routing middleware in providing the large data sets required for dynamical systems analysis (DSA). Via DSA the

system's stability, or instability, can be verified at an early stage, aiding a system designer at the prototype phase. The middleware provides three main types of information to the DSA:

- Sensor Data / Generated Context
- Behaviour exhibited by the context-aware system i.e. Actuators
- Actual Events i.e. the delta between the simulated sensor data and a full picture of the deployment environment.

The candidate environmental monitoring application used as the example in this paper is a temperature moderation system. The goal of the system is to maintain the temperature of a room at a comfortable level for the room occupants. The system has access to an actuator on a window, and it can open the window at various widths; the second control that the system can access is the radiator in the room. The smart space also consists of a temperature sensor placed in the room.

This paper begins in the next section by discussing some background and related work. Sections 3 and 4 present the infrastructure behind this work in terms of the simulation, modelling and application interface, as well as the requirements for DSA. Section 5 discusses the dynamical systems analysis approach followed by some conclusions about the paper in section 6.

2. Background

Ongoing problems of the cost and effort required to create ubiquitous computing (ubicomp) test environments continue to be barriers to large scale testing in a rapid prototyping cycle. Additionally complexity in context-aware ubicomp systems builds rapidly due to the multitude of contextual inputs sourced from users, devices and the surrounding environment. Heterogeneity of contextual information and scalability of ubicomp systems are repeatedly listed as major challenges for ubicomp developers, Pham et al [16] list these as the primary challenge. Research is also indicating that context does impact on end user acceptance of systems; Potts et al [17] have completed work using the Inquiry cycle which has already identified that an individual's location impacts on that user's acceptance of a service.

2.1 Environment Monitoring

Environmental sensing focuses on monitoring events or phenomena in the real world; these include the temperature in a room, the humidity in a space, or contaminants in a lake. Often environmental monitoring will place sensors in a remote location, in these circumstances the sensor is often used to detect an event that can, should it occur, cause catastrophic failure of the sensor. In [18] a context-aware system of environmental sensing for forest fires is proposed. Pesch et al put forward a method whereby a node's health is reflected by the temperature that node currently detects. A linear degradation of the 'health status' occurs between 0 and 100. If a node's health is 0 it is deemed likely to fail in a short period of time, in this circumstance the system as a whole reconfigures to ensure connection can be maintained. Analysis of multiple simulated scenarios can aid in determining system criteria, e.g. sensor location.

2.2 Simulation

Morla & Davies' [11] simulation tool focuses on addressing many of the same issues as the platform described in this document, however without a visualisation element. Morla and Davis use existing simulators to rapidly and cost effectively evaluate

healthcare applications in terms of their networking performance and ability to process GPS information. Reynolds et al [12] address the issue of ubicomp system design from the ground up. Their work focuses on an automated approach to simulation and successfully models four abstractions of ubiquitous computing scenarios: the environment, sensors, actuators and applications. Their simulation environment supports an automated approach by using a 2D grid tailored towards playing out traffic scenarios.

2.3 Context Aware Applications

2.3.1 Design

A central issue for evaluating ubicomp design is to show that the presented solution is appropriate and that serves its intended purpose as was stated by Nuseibeh in [19]. However the criteria that must be satisfied for 'good' ubicomp design is yet to be standardized. Some researchers choose their own set of metrics to determine that a solution is successful in its appropriateness while others deem that reusability is a measure of success. Davies [21] points this out, stating that the lack of comparison across ubicomp system design is a roadblock to progress. Davies lists the development of comparative metrics for ubicomp applications as one of four high priority issues to improve the pace moving forward.

2.3.2 Prototyping

The context toolkit [20] was developed to make prototyping of context aware applications easier and faster by using context widgets. Context widgets offer a high level method for accessing contextual information so that the application does not have to be concerned with individual sensor and protocol implementations. Widgets provide the benefits of abstraction, reusability and they hide the underlying complexity of the sensor network. The context toolkit proved useful both in creating new ubicomp applications and also helped retrofit existing applications to support context-awareness.



Figure 1. Simulated Environmental Research Institute (ERI), University College Cork

2.4 Dynamical Systems Theory

Dynamical systems theory can be used to determine whether a system is random, chaotic, periodic, or stable. Determining this for a complex system is often an intractable problem in linear

mathematics. Dynamical systems theory is an approach, which has been shown to produce results in a multitude of domains [3][4]. When designing a system, knowing how the components interact can aid in a number of ways. The types of interactions include periodic stable (the system repeats a pattern over a given period), chaotic (chaotic systems are systems which display a high degree of sensitivity to initial conditions), or random (the system shows little correlation between the inputs and outputs). The application of dynamical systems theory to context-aware computing systems was previously proposed in [7], however a lack of large context-aware data sets has to date been problematic for the work.

3. Simulations and Modelling

As mentioned in the introduction, 3D simulators are particularly good at generating context related to location and spatial relations; in this work we use the Pudecas simulation environment, figure 1. The role of the Pudecas simulator is to enable the middleware to supply the DSA with contextual information about the test environment. This contextual information is generated by virtual sensors in the same manner as real sensors i.e. by (virtual) user activity in the test environment. The context routing system (CRS), which will be discussed in section 4, provides for two way asynchronous communication between the context aware application and the test environment. This two way communication is required to test environment monitoring applications in this manner. These types of applications exhibit their behaviour as actuations in their environment; the CRS provides the mechanism for these applications to instruct the virtual environment.

The execution phase of an experimental cycle allows for either multi-player or single user, bot populated experiments. Multiplayer simulations allow up to 32 users to experiment with the System Under Test (SUT) simultaneously in the context of the virtual world. Bot driven simulations on the other hand involve a single user testing the service while role playing bots also roam the virtual world testing defined scenarios. During either user-driven or bot-driven experiments, sensor data is generated and sent to the middleware. Bots are particularly useful for experiments of long duration because they can be left unsupervised for prolonged periods or overnight.

Simulation in this work allows:

- Generation of simulated sensor data at runtime, based on the activity of real users and non-player characters.
- Actuations of entities in the virtual world e.g. switching on lights, opening doors. These actuations happen when signalled by the SUT.
- Ease of sensor deployment and reconfiguration.

The architecture and supporting toolset for Pudecas is discussed in more detail in O'Neill et al's earlier publication *A Simulation-based Approach to Highly Iterative Prototyping of Ubiquitous Computing System* [22].

3.1 Airflow Modelling

The Pudecas simulator is based on the Source/Half-Life 2 game engine [5]. Game engine technology already models and generates the three dimensional physical space. However a key weakness for these types of simulators is in testing applications such as the environment monitoring system presented in this

paper. Some success has been seen using the Half-Life 2 engine to conduct fire simulations at Durham University [13], however for most environmental conditions additional modelling must be provided. Work has commenced on the Pudecas simulator [22] to introduce a 'rule of thumb' airflow model. Building on the existing sophisticated calculation and rendering tools available through the Half-Life 2 SDK, a model of airflow is being built on top.

The inputs to this model include:

- a) Wind Direction
- b) Built Environment Geometry
- c) Air particle density (no. of air particles) [default 1/metre]
- d) Wind velocity [default 4 m/sec]
- e) Edge conditions of model space (default boundaries as building elements)

The visualisation element included in the model is for the designer's benefit. Air particles are used to allow tracking of the airflow and rules have been written to determine the direction of the airflow, for example:

- Air particles follow wind direction in free space.
- Air particles adjust position and direction to maintain equal spacing between different air particles and enclosing built elements (e.g. constriction of space by converging walls)
- Air particles are influenced by adjacent air particles: velocity increases with proximity of neighbouring air particles /building elements (and vice versa, they slow with increasing distance)

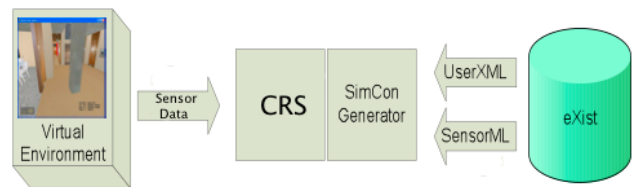


Figure 2. Gives a high level view of the flow of context from the Interactive Context Generator to the SUT.

3.2 SimCon Modelling

Data generated by a game engine is highly accurate, it is possible to know the exact location (within a Euclidean geometry) and orientation of a user at all times, and all location updates are synchronised to the systems global clock. This is an unrealistic level of accuracy for the real world but with this information it is theoretically possible to simulate a wide range of different sensors. By introducing Gaussian noise and delays to the context flow generated by the virtual environment, a more accurate representation of the types of context provided by low level context sources can be generated.

The SimCon modelling tool supports rapid experimental configuration by generating SensorML descriptions which are used at runtime to produce the context flow supplied to an application under test. On start-up, the SimCon Generator loads in all sensorML context source descriptions, see figure 2, to produce a more realistic context flow for testing and experimental purposes. SimCon provides efficient and accessible tools to place

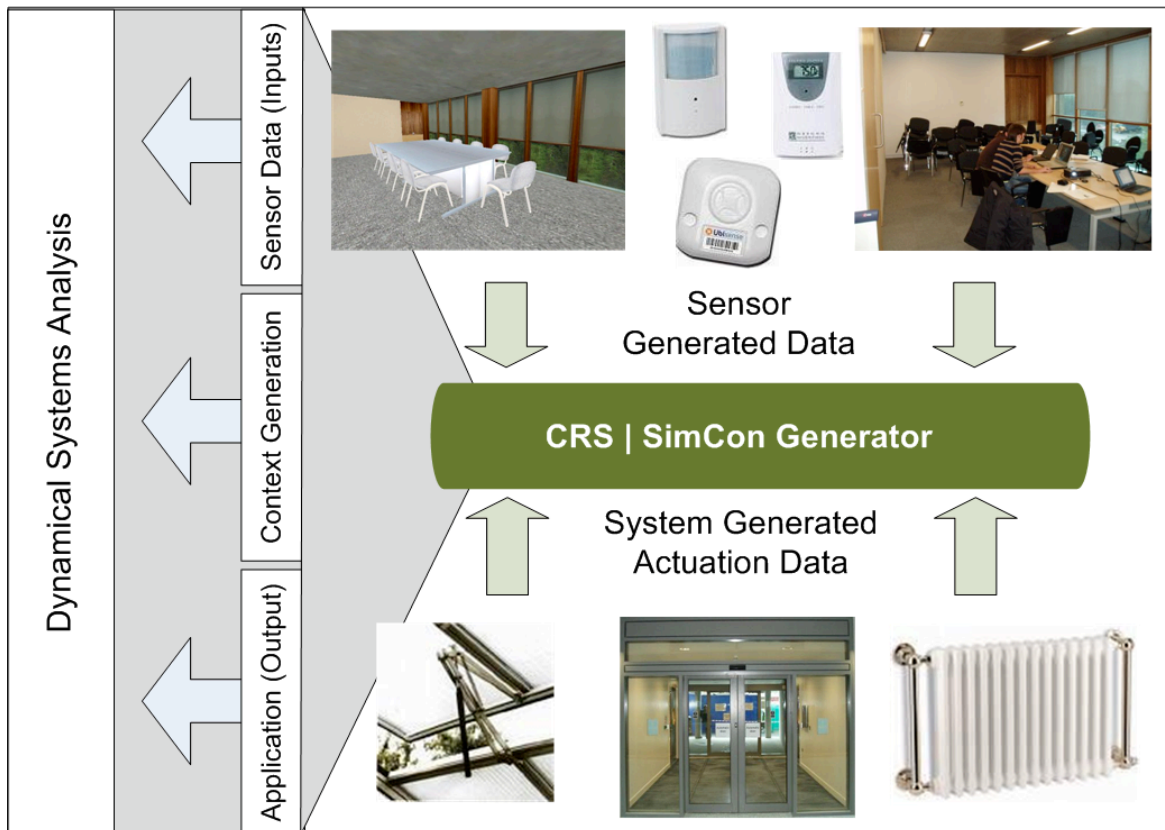


Figure 3. System Integration

and configure differing location based context sources within the virtual environment, providing heterogeneous types of location based context to evaluate their impact on the SUT. SimCon is described in detail in McGlenn et al's paper *Modelling of Context and Context Aware Services for Simulator Based Evaluation* [23].

3.3 DSA Requirements

Enabling dynamical systems analysis for context aware applications requires significant volumes of data that includes the inputs to the context-aware system and the behaviour of the smart space. In a real world test environment it is difficult to generate these large data sets while also monitoring a large system and all of its users. A simulation based approach provides the affordance of capturing a full picture of actual events as well as the snapshot generated by the virtual sensors, which enables DSA to analyse the system output against the input and against actual events. DSA can perform better if problematic behaviour can be connected to specific outputs from the analysis, thus enabling greater confidence in the results. Work is being conducted by O'Neill et al [22] towards automation identification of potentially problematic behaviour in a context aware system.

4. Context Routing

In figure 3, a high level view is shown of the system integration. At run-time, messages flow between the virtual environment and the context aware service. As mentioned in section 3.2, SimCon models data leaving the simulator so that it becomes the contextual information on which services base their decisions and thus respond to the user's needs. Context is routed to applications

from the test environment through a Java based context routing system (CRS). This offers a number of advantages:

- Context aware applications designed to run on real sensors do not have to be changed for testing in the virtual environment. The Java system is responsible for meeting the subscription requirements of an individual application. This can be in terms of the sensor data supplied or the data format. Currently XML is used for data exchange
- The CRS manages queuing of messages, taking the burden from both the simulator and the application under test.
- Applications can be seamlessly moved between the real and virtual world without being recoded. The simulator has proven useful for early stage testing, especially when working with more sophisticated technologies such as Ubisense where configuration of the system requires significant effort.
- The test platform can be distributed; applications can run on mobile devices or emulators.
- The CRS can capture data that the application does not have access to. For the purpose of fidelity, applications only receive information from the environment which they would receive if it was a real environment. However the CRS can gather additional data specifically for the DSA approach so that exhibited behaviour can be analysed against *actual* environment conditions, not just *sensed* environment conditions.

A configuration file provides the necessary information for an experiment to begin e.g. an experiment ID, a map name, a game-server address and data subscription information. This approach allows other applications to subscribe to a single experiment. This is particularly important for performing analysis; the DSA tool must be able to subscribe to the same context flows and environment as the application in order to perform its analysis.

Once the service is registered, the CRS invokes a new game-server on the remote host and subsequently establishes a connection with the simulation for experimental data transfer. During an experiment, services may connect to or disconnect from the proxy, thus joining or leaving experiments respectively. Services send asynchronous instructions to alter the state of the environment through device or entity actuation, e.g. opening a window or altering the thermostat settings. The latency of messages, across the system is of the order of milliseconds.

5. Dynamical Systems Analysis

In dynamical systems theory a system is composed of an ambient space, sometimes called a 'state space'. This ambient space is a multi-dimensional construct, whereby each input and output is a dimension of the system. Consider the smart space previously described, within this smart space the detected temperature, and the speed of the airflow can both be taken as inputs and the amount by which the window is opened is an output. If the room is too warm, the window needs to be opened. However, in order to prevent the room from dropping in temperature excessively, the speed of airflow must also be considered; if there is a strong breeze, the window will need to be opened less than if there is a gentle breeze.

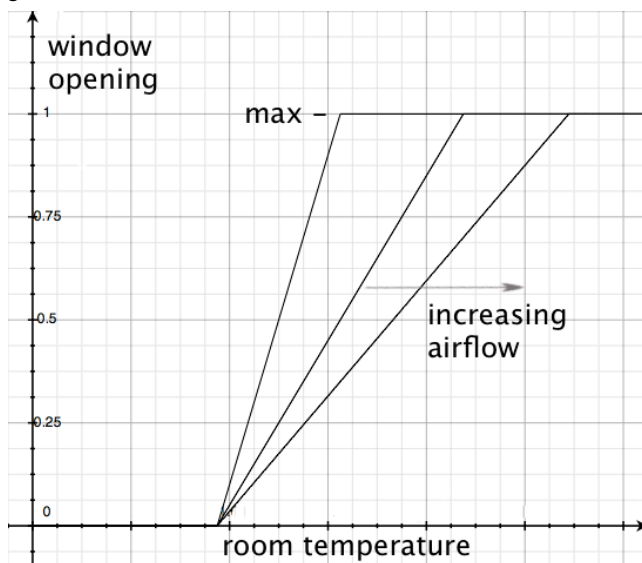


Figure 4. Effect of Airspeed on Window Actuator

Figure 4 displays this information in a 2-dimensional manner, however the actual ambient space exists in multiple dimensions and consists of a surface on which the system should remain. The determination of the optimum surface can require domain specific knowledge, however DSA can aid in the verification of these surfaces. Dynamical systems can have a number of geometries present in their ambient space. These include point attractors, strange attractors, and periodic trajectories. For all of these geometries, there exists a 'basin of attraction'; if the system enters

a state that is within this 'basin of attraction' it will converge to the respective attractor.

Also of interest in smart spaces is the level of stability, this can be determined by a systems Lyapunov exponent [8]. Effectively this value represents whether a system will converge to a stable solution, and if so, how quickly this convergence takes place. For the purposes of this paper, all configuration is viewed from the perspective of the application developer. In other words, it is assumed that the smart space has already been installed and exists in a given setup.

There exists a quantitatively intractable problem within the temperature control system outlined in this paper, and that is the determination of the accuracy of a given application logic; for example is a given application's logic able to deal with outlier circumstances, and does the system as a whole ever enter into states or areas of the ambient space from which it cannot escape without an external stimulus?

The dynamical systems analysis component of the system, accepts inputs from the middleware. These inputs will be values that vary in time for the given system. In the given example of a window-temperature control system, the inputs would include the average ambient temperature in the room, the temperature as given by a temperature sensor, the position of the window (open or closed), the temperature outside the room, the status of the heating in the room, the applications decision in the context of all this data (for example, to open the window, close the window, decrease/increase the heating settings).

For a given set of context from a smart space, dynamical systems analysis can determine if there is a correlation between inputs; a strong correlation may indicate redundancy in the sensor configuration, or if a system is highly sensitive to initial conditions (moving the system into the realm of chaos theory), or indeed if the system has a tendency to enter into a steady-state of its own accord.

For a given application logic, coupled with the context data, dynamical systems analysis can aid in determining if the logic applied by the developer moved the system into a steady-state, a chaotic state, or a random state. Following this, the application developer can alter the logic in the application, and the same simulation can be executed again. The benefits of simulation now become apparent; it enables the applications logic to be compared against the same context, while also enabling the injection of outlier situations.

6. Conclusions

Early stage testing of context-aware systems is hampered by limited availability of deployed sensor rich environments. In addition to the challenges of mobile and distributed system development, context aware systems are further complicated by adaptivity in their exhibited behaviour. Rapid system development, early in the design cycle, requires a middleware that enables applications to interface with an easily configurable, test environment. In this paper we have discussed the role of a context routing and context modelling software which allows prototype context aware systems to benefit from a 3D simulated test environment. Simulation enables quick and easy deployment of sensor rich environments, enabling large-scale dataset creation.

Analysis of these datasets, using DSA, can give application developers access to qualitative predictions regarding the system, and thus help to ensure that the application will remain within

bounds deemed by the developer to be optimum, even when that system cannot be solved analytically. Simulation simultaneously allows an application to be tested against long-run average scenarios and outlier events. These results can be used to reconfigure the system and/or environment (i.e. the simulation) leading to improved system design. The system is not a replacement for real-world testing, however the system should aid rapid prototyping of systems at early stages.

7. Acknowledgements

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) and by the HEA funded NEMBES project (www.nembes.org).

8. References

- [1] Li, Y. Hong, J. I. and Landay, A. Design Challenges and Principles for Wizard of Oz Testing of Location-Enhanced Applications. *IEEE Pervasive Computing* 6, 2 (Apr. 2007), 70-75.
- [2] H. V. D. Parunak, R. Savit, R. L. Riolo, and S. J. Clark. Dasch Dynamic analysis of supply chains. Technical report, CECERIM, 1999. Final Report.
- [3] E. Stone and S.A. Campbell, Stability and Bifurcation Analysis of a Nonlinear DDE Model for Drilling. *Journal of Nonlinear Science*, Volume 14, Number 1, pp27-57
- [4] H. Poincare, Sur le probleme des trois corps et les equations de la dynamique (), *Acta Mathematica* v.13 (1890), pp1-271
- [5] Half-Life 2 <http://www.valvesoftware.com/games.html>. Source Engine <http://source.valvesoftware.com/>
- [6] George Osipenko, *Dynamical Systems, Graphs, and Algorithms*. LNCS 1889. Springer, 2006
- [7] Simon Dobson, Eoin Bailey, Stephen Knox, Ross Shannon and Aaron Quigley, "A first approach to the closed-form specification and analysis of an autonomic control system", 12th IEEE International Conference on Engineering Complex Computer Systems, Auckland, New Zealand, July 11-14, 2007
- [8] P. J. Moylan and D. J. Hill. Stability criteria for large-scale systems. *IEEE Transactions on Automatic Control*, AC-23(2):143-149, April 1978.
- [9] Barton, J. and Vijayaraghavan, V., "UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment", <http://www.hpl.hp.com/techreports/2002/HPL-2002-303.html>
- [10] Li, Y., Hong, J. I., and Landay, J. A. 2004. Topiary: a tool for prototyping location-enhanced applications. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA, October 24 - 27, 2004). UIST '04. ACM, New York, NY, 217-226. DOI= <http://doi.acm.org/10.1145/1029632.1029671>
- [11] Morla, R. and Davies, N. 2004. Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment. *IEEE Pervasive Computing* 3, 3 (Jul. 2004), 48-56. DOI= <http://dx.doi.org/10.1109/MPRV.2004.1321028>
- [12] Reynolds, V., Cahill, V., and Senart, A. 2006. Requirements for an ubiquitous computing simulation and emulation environment. In *Proceedings of the First international Conference on integrated internet Ad Hoc and Sensor Networks* (Nice, France, May 30 - 31, 2006). *InterSense '06*, vol. 138. ACM, New York, NY, 1. DOI= <http://doi.acm.org/10.1145/1142680.1142682>
- [13] S.P. Smith and D. Trenholme, [Rapid prototyping a virtual fire drill environment using computer game technology](#), *Fire Safety Journal*, Elsevier.
- [14] Cerpa, A., Elson, J., Hamilton, M., Zhao, J., Estrin, D., and Girod, L. 2001. Habitat monitoring: application driver for wireless communications technology. In *Workshop on Data Communication in Latin America and the Caribbean* (San Jose, Costa Rica). SIGCOMM LA '01. ACM, New York, NY, 20-41. DOI= <http://doi.acm.org/10.1145/371626.371720>
- [15] Bandini, S., Mosca, A. and Palmonari, M. A Hybrid Logic for Commonsense Spatial Reasoning. *AI*IA 2005, LNCS*, vol. 3673, pp 25-37.
- [16] Pham, H. N., Mahmoud, Q. H., Ferworn, A., and Sadeghian, A. 2007. Applying Model-Driven Development to Pervasive System Engineering. In *Proceedings of the 29th international Conference on Software Engineering Workshops* (May 20 - 26, 2007). ICSEW. IEEE Computer Society, Washington, DC, 193. DOI= <http://dx.doi.org/10.1109/ICSEW.2007.43>
- [17] Potts, C., Takahashi, K., and Antón, A. I. 1994. Inquiry-Based Requirements Analysis. *IEEE Softw.* 11, 2 (Mar. 1994), 21-32. DOI= <http://dx.doi.org/10.1109/52.268952>
- [18] Bernd-Ludwig Wenning, Dirk Pesch, Andreas Timm-Giel and Carmelita Görg. Environmental Monitoring Aware Routing in Wireless Sensor Networks. *IFIP International Federation for Information Processing*, Volume 284/2008, pp. 5-16
- [19] Nuseibeh, B. and Easterbrook, S. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM, New York, NY, 35-46. DOI= <http://doi.acm.org/10.1145/336512.336523>
- [20] Salber, D., Dey, A. K., and Abowd, G. D. 1999. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit* (Pittsburgh, Pennsylvania, United States, May 15 - 20, 1999). CHI '99. ACM, New York, NY, 434-441. DOI= <http://doi.acm.org/10.1145/302979.303126>
- [21] Davies, N. Proof-of-concept demonstrators and other evils of application-led research: A position statement. In *Proceedings of UbiApp Workshop*, Munich, Germany, 2005. www.cl.cam.ac.uk/ubiappswweb
- [22] Eleanor O'Neill, David Lewis, Owen Conlan, A Simulation-based Approach to Highly Iterative Prototyping of Ubiquitous Computing System, *SIMUTools 2009: Proceedings of 2nd International Conference on Simulation Tools and Techniques*, Rome, Italy, 2-6 March, 2009
- [23] Kris McGlinn, Eleanor O'Neill, David Lewis. Modelling of Context and Context Aware Services for Simulator Based Evaluation, *MUCS 2007, 4th International Workshop on Managing Ubiquitous Communications and Services*, Dublin, Ireland